AD-A259 004

AFIT/GA/ENY/92D-15

DATA REDUCTION WITH LEAST SQUARES DIFFERENTIAL CORRECTION USING EQUINOCTIAL ELEMENTS

THESIS

Michael Scott Wasson
Captain, USAF

AFIT/GA/ENY/92D-15

DTIC
ELECTE
JAN 0 8 1993
S B D

93-00143

Approved for public release; distribution unlimited

93 1 4 140

AFIT/GA/ENY/92D-15

DATA REDUCTION WITH LEAST SQUARES DIFFERENTIAL CORRECTION USING EQUINOCTIAL ELEMENTS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Space Operations

Michael Scott Wasson, B.S.

Captain, USAF

December, 1992

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Approved for public release; distribution unlimited

DTIC QUALITY INSPECTED 1

Preface

The art of orbit estimation is a fairly new interest of mine. My initial assignment in the Air Force as an orbital analyst spawned my interest in this subject. My background is primarily based in math and computer programming, but the deeper I find myself in orbital mechanics and the accompanying theories, the more I know that my future interests will be in the astrodynamics field.

This study may not be overly complicated or all that difficult to the practiced orbital mechanic, but my personal knowledge gain is invaluable. I feel that the programming I completed building the truth model and writing the differential corrector has given me a solid base of knowledge for continuing my interests in this field.

I would like to recognize Dr. Wiesel for the challenges he presented while teaching the theories of astrodynamics and orbit determination. This thesis is specifically derived from one of his ideas. I appreciate his overwhelming patience as I ever so slowly gained confidence in my astrodynamic abilities. Dr. Kelso was also invaluable with his guidance in programming and total emotional support. I would also like to thank my fellow student, Capt Jeff Berger, for answering all of my "stupid" questions and keeping me on the right track early in the theoretical development.

Above all else, I thank my wife Amy for her undying patience and understanding. Her constant encouragement allowed me to finish this project with a true sense of accomplishment.

Michael Scott Wasson

Table of Contents

List of Figures

List of Tables

Abstract

This study investigates earth satellite orbit estimation on a track of range, azimuth, and elevation data from a single tracking station. The estimation routine is a least squares batch filter based solely on two-body orbital motion. Using equinoctial elements for the reference orbit avoids the numerical difficulties of the classical elements at eccentricities near zero and inclinations near zero or 90 degrees. Orbits for Mir, DMSP, Explorer, Cosmos, and GPS are investigated. The goal of this study is to reduce orbit information from observations (range, azimuth, and elevation) to an element set and a covariance matrix without considering perturbation effects. The results indicate that the lower orbiting earth satellites had large J_2 perturbations on the equinoctial elements causing the differential corrector to diverge. Higher orbiting satellites had minimal J_2 effects and the correction process sufficiently extracted all information from the data and successfully reduced the observations to an element set and a covariance matrix.

DATA REDUCTION WITH LEAST SQUARES DIFFERENTIAL CORRECTION USING EQUINOCTIAL ELEMENTS

I. Orbit Estimation

1.1 Introduction

Modern space technology provides the means for tens of thousands of satellites to orbit the earth. As a result, there is a basic need for tracking these satellites and for predicting the orbital paths they follow. The military organization established to track and catalog orbiting space objects is located within Cheyenne Mountain in Colorado Springs, Colorado. One of the primary missions of the North American Aerospace Defense (NORAD) Command is to keep track of every orbiting object, whether it is operationally functional or a piece of space debris. The Space Surveillance Center (SSC), located within NORAD, processes this positional information.

The data needed to estimate an object's orbit comes from one of several different methods: direct radio link contact, echoing a radio signal off of the structure of the object, or via optical observations. Various remote tracking stations located throughout the world gather the positional information data described by a set of six orbital elements. Typically, the six elements consist of a position and velocity vector (three elements are needed for each) or tracking station RAER data (range, azimuth, elevation, and range rate).

Low-earth orbits, within a band of altitude from 100–1000 kilometers, represent a majority of satellites. Low-orbiting satellites have much higher velocities and thus pass over tracking stations in 10–30 minutes. These stations collect several "observations" (position and velocity information) within this time and transmit the data to the SSC. For a particular orbiting object, there may be 10 to 15 passes over the same station in a day producing 10–1000 observations. The immense data

processing load of tracking over 7000 objects inhibits NORAD's ability to accurately track each one.

The number of observations collected for a particular object is not individually overwhelming. However, it is the collection of observations for several different orbits over time that increases the processing load. Independent orbit estimation at individual tracking stations gives flexibility in prioritizing satellite contacts. The resources are then available for accurate tracking and location of a greater number of objects.

1.2 General Issue

Data compression of the orbital position and velocity information into an element set and a covariance matrix reduces processing time within the SSC. Current orbit estimation theory techniques are available at the tracking stations for on-site estimation. It is possible for each site to perform the bulk of the initial computations by extracting all information about the orbit along with typical deviations from the theoretical path (perturbations). The site sends a single element set describing the complete arc of data over the station along with a confidence level (how accurately the station believes the element set describes the actual orbit) to the SSC. The processed data is in the form of an element set instead of the actual position and velocity data (or the RAER data).

By shifting the initial estimate responsibility to the tracking stations, it is possible to collect an increased number of observations for each satellite pass. The greater data load occurs only at each site, keeping the SSC free from the burden of initially estimating orbits for thousands of objects. This leads to a more accurate estimate of each object in the catalog which, in turn, allows for less frequent orbit updates on that object. The overall effect is a much more accurate catalog of orbit information.

1.3 Background

Orbit determination at the tracking sites uses currently available theories and methods. It shifts the initial estimation responsibility from the SSC to the tracking stations. The goal is not to accurately describe the whole orbit of the object, but to extract as much information as possible out of the observations and reduce them into a more manageable element set and a covariance matrix. With an increasing number of objects requiring orbit estimation, the SSC can not afford to collect thousands or even hundreds of data points for each one. This would ultimately lead to a sub-standard element set as a final product. Over time, either the object is going to need an update to its element set more often, or it will degrade past the point of being able to be tracked at all.

The equinoctial elements (see Appendix A) have desirable numerical stability properties. Because all elements, except the mean anomaly term, are constant, it is theoretically easy to determine an estimate of the state vector. The main challenge is to converge upon a solution that will describe a whole orbit based on only a few minutes of data. The achieved orbit is not operationally accurate as a stand-alone estimate and is combined with several other element sets to give the best final orbit estimate possible. Constant equinoctial elements for the two-body problem simplifies the original estimation and several element sets processed together provide the means for identifying the perturbations.

1.4 Problem Statement

The future of accurate orbit estimation at NORAD depends on new data reduction techniques to handle the 7000+ orbiting objects. Preliminary analysis at the data collection sites using a least squares differential corrector based on two-body motion is one possible technique for reducing the data into an element set and a covariance matrix. Stable equinoctial elements simplify the analytical computations for building the estimation model.

1.5 Research Objectives

Research into the data reduction problem involves three main objectives. The first step involves building a truth model based on two-body orbital motion, the J_2 zonal harmonic, and air drag producing accurate range, azimuth, and elevation observations for several tracking site locations. The truth model adds Gaussian random noise to the "perfect" data and then organizes the output into separate passes. Secondly, a differential corrector propagates an equinoctial reference state to the first observation time and then iteratively solves for a state correction and a covariance matrix based on two-body equations of motion and least squares batch methodology. Finally, the new differential corrector estimates a new state vector based on truth model data for five representative orbits. Unique characteristics from each orbit case expose any possible deficiencies in the model.

1.6 Research Questions

The least squares method is a proven technique for accurate differential correction. However, because it is a batch estimator, large amounts of data are required to mathematically solve the orbit problem. Low-earth orbiting satellites have periods near 90 minutes with passes of only 10–30 minutes. The short arc of data may not contain enough observations to allow least squares to converge to a solution. Included in the intent of this study is the examination of the differences between low and high-orbiting satellites to determine if the pass length is a crucial factor in the differential correction of data from one station.

When limited to the two-body problem, the only time-varying element is the mean anomaly — all other elements remain constant. Only two-body element sets are derived through orbit estimation at the sites because there are not enough observations on the low-earth satellites to determine any perturbation effects. Therefore, actual perturbations in the data may cause problems with convergence. Other element sets, such as the classical orbital elements, have numerical instabili-

ties for earth-orbiting satellites, so the next step is to use the canonically transformed equinoctial elements.

1.7 Assumptions

The following assumptions are critical to the development of the methods examined in this thesis:

- Because of the short interval time of data collection from the remote tracking sites, there is not an adequate amount of data to accurately predict any perturbation effects on the satellite. However, these perturbations do exist. There is an underlying assumption that enough information about the orbital track can be extracted and placed into an element set to accurately describe the track that it was taken from.
- In order to simplify the mathematics of the data linearization matrix, H, a two-body estimator is used. H is still mathematically intense (see Appendix D), however, it does have a closed-form solution when perturbation effects are left out. The model formulation assumes that the two-body estimator is sufficiently accurate to obtain a solution.
- The truth model contains accurate ephemeris information for the input satellite reference trajectory. This data is assumed to be "perfect" and made more realistic by applying Gaussian random noise.
- Solutions to the orbital problem have errors associated with the dynamical system, the state vector, and the observations. The foundation for the estimation routines developed here assumes that the only source of uncertainty in the model lies in the observations.

1.8 Scope

The proposed study models the effects of applying a two-body least squares batch estimator to a short arc of data to create an element set. This examination is being conducted solely on

the perceived current procedures of Cheyenne Mountain and its dedicated ground stations and will not address any scheduling changes or procedure changes other than sending an element set rather than the actual observations to the SSC.

The main effort in this thesis is writing a truth model to generate earth tracking station satellite observation data (ephemerides) and writing a least squares differential corrector based on the equinoctial element set. Once completed, the analysis of the estimator performance is examined for low and high-altitude earth orbits. This encompasses the intent of the study.

1.9 Summary

The orbit determination problem has evolved over many centuries. Throughout this time there has been a common thread among orbital analysts to achieve the most accurate description possible of an orbit using a wide variety of available techniques. For example, one technique performs the initial estimation at the remote tracking site based on a single arc of data. This arc describes only a small portion of the entire orbit for low-altitude satellites and larger portions for higher altitude satellites. Extracting all the orbit information contained in the arc reduces several observations into an element set and associated confidence matrix (the covariance matrix). The equinoctial elements provide a solid basis for determining a two-body solution and reducing the data into the desired element set. Allowing the individual tracking stations to reduce the data establishes a more accurate catalog of orbits reducing the number of orbital updates per object which, in turn, allows the pass scheduling of a greater number of objects. This study examines the use of equinoctial elements as the basis for a two-body differential corrector for a single arc of satellite observation data.

II. Literature Review

2.1 Introduction

Making use of onboard payloads such as navigation, weather, or early warning, usually requires accurate orbit prediction using orbit determination methods that solve for parameters which completely specify the motion of the satellite. A differential correction process takes a theoretical (or reference) orbit path as the starting point and makes adjustments to account for perturbations and measurement noise. This process must be repeated periodically to determine the best possible "fit" of the observations. Two-body motion with constant classical elements of $\{a, e, i, \Omega, \omega\}$ and a time-varying mean anomaly, M, describe the principal reference trajectory. Both batch and sequential filters, individually or in combination, accomplish orbit determination using probability theory and matrix manipulation.

2.2 Batch versus Sequential Data Processing

Modern orbital estimation theory consists of batch and sequential processing methods. Batch processing cannot begin until a group (or batch) of data arrives. A batch of data consists of several observations (position and velocity values), possibly from several tracking stations. The manipulation of a batch of data allows for trend analysis but requires a large amount of storage space. Sequential estimation, as its name implies, processes individual observations sequentially as they arrive. The advantages of sequential methods are speed and ease of data handling.

2.2.1 Batch Estimators. Batch estimation algorithms are the foundation for all orbit determination schemes available today. This estimation technique processes a large batch of observations giving a new state vector as its product. By processing many data points at the same time, it is possible to obtain an orbit that passes "close" to all data points. This classical estimation theory centers on deterministic dynamics. However, with errors in the data due to imperfect instruments,

the solution methods employ probabilistic methods. The method of least squares is a batch estimation method and is discussed in Section 2.3.

2.2.2 Sequential Estimators. Sequential estimation replaces the deterministic dynamics of the batch estimator with a stochastic process. Statistical behavior now characterizes both the observations and the dynamics. The estimation algorithm extracts information about the initial position of the object being tracked from observations with errors. The estimator is called a filter and errors in the data are called noise.

A sequential filter, such as the minimum variance method, uses statistical filtering to determine the best possible estimate of a state vector based on deviations of the observations from the reference trajectory. Obtaining a good estimate of the orbit assumes and uses errors in the observations and the state vector. So far, this describes both batch and sequential filters, but the basic difference between the two is that the sequential method continuously updates the orbit with each new observation whereas the batch method receives all data before processing (7:23-24).

A Bayes filter is a sequential estimator. This filter works almost identically to a batch estimator except that the required inputs are an initial trajectory with covariance and *new* data. Sequential estimators do not require all of the data points ever produced to extract a solution. It is possible to use the estimate obtained in a sequential filter as data input to another estimator. Instead of using the old data points, the old estimate holds all the important information about the edited observations. Sequential filters allow for greater computational efficiency because of the smaller data sets (15:90).

Kalman filters are mathematically identical to Bayes filters but expressed in a different form. The estimation process of a Kalman filter works especially well for very small numbers of new data points. The primary advantage is speed, but the tradeoff is the increased chance that the sequential estimator will not accurately describe the trajectory. The ease of adding new data to the estimate is a desirable quality of a Kalman filter (15:99).

Both the Bayes and the Kalman filters have a definite advantage over batch methods when inverting the covariance matrix. The number of parameters being estimated dictates the size of the matrix inversion in a batch estimator such as least squares. Minimum variance, and other sequential methods, invert a smaller matrix based on the number of observations processed at a time. By processing each observation separately, the inversion is trivial. This process also allows for greater flexibility because the process can be stopped at any point and the current state of the estimation process is an estimate. Also, discarding each observation after processing reduces the amount of required storage space (7:24).

2.2.3 Tradeoffs. Both batch and sequential methods are non-linear estimators. In each case, the estimation process works by linearizing the equations of motion about a reference trajectory. Keeping low-order terms (usually just first order) in a Taylor series expansion simplifies the equations.

If the dynamics of predicting a satellite's position were purely stochastic, the methods of batch and sequential estimators would be equivalent. Nevertheless, orbital systems are deterministic in nature, creating differences in the two methods. A low-orbiting satellite that is affected by air drag shows how the two estimators differ. Batch estimators locate possible errors in the data due to drag because they have all of the data available over a given time span. Errors in the data mislead a sequential estimator over short time periods, thus causing more difficulty in estimating the desired air drag effect. This leads to uncertainty in the data and makes the air drag component effectively unobservable (15:95–97).

Current Space Surveillance Center procedures take advantage of the speed of sequential estimation methods to process the data. Yet, sequential estimation methods alone cannot be used to accurately describe the orbital trajectories being estimated. A batch estimator gives a good overall look at the trajectory but are not always used because they are extremely time consuming and cannot handle thousands of objects.

2.3 The Method of Least Squares

The method of least squares was devised by Karl Friedrich Gauss as early as 1795 (9:164). The first publication of the least squares method, however, was by Legendre in 1806. Legendre used least squares while trying to calculate the orbits of comets. Throughout the seventeenth century, mathematicians such as Pascal, Fermat, and Bernoulli developed probability theory (11:102–103). In 1809, Gauss used the probability theories as a basis for least squares and in 1810 published the symbols and notation that are still associated with it today (6:14).

Least squares is the foundation for estimation theory. This method assumes that the dynamics contain no errors (deterministic), but the observations do contain errors. Therefore, information on the dynamical system must be extracted from the imperfect data. Least squares tries to obtain an orbit which passes as close as possible to all of the data points (observations) (15:56).

The estimation process is based on the Central Limit Theorem which states that all errors associated with the data will be distributed following a normal, or Gaussian, distribution (see Figure 1) (15:8). This curve was first described in 1733 by De Moivre but Gauss extensively used the theory behind the curve while making astronomical observations and measurements in geodetic surveying (11:104).

By examining the curve, larger errors occur with less frequency while smaller errors are more likely to occur. The positive errors tend to cancel out the negative errors (11:104–105). The curve is of the form

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{1}$$

where $P(x)$ is the probability of a given x to occur. The value in the exponential term is the most critical for maximizing the probability of a value of x to occur, because the leading terms in the

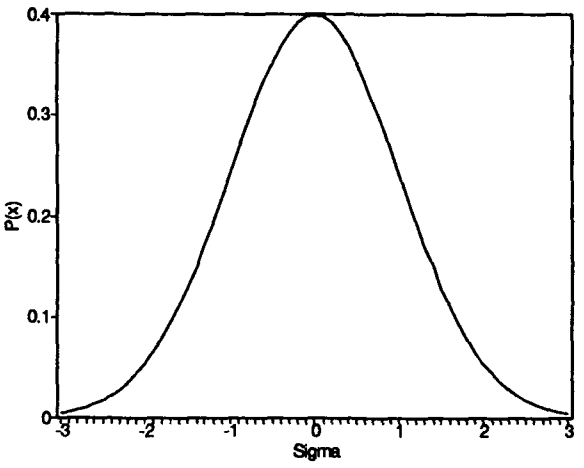


Figure 1. Gaussian Distribution

equation are constant. The exponential term can be rewritten for all x as

$$-\sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{2\sigma_i^2} \tag{2}$$

where $\bar{x}$ is the *average* estimate of x for all N observations and σ is the ***standard deviation*** of each measurement from the actual value. Maximum probabilities are achieved by minimizing the positive exponential expression. This is a straightforward calculus minimization:

$$\frac{d}{d\bar{x}} \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{2\sigma_i^2} = 0 \tag{3}$$

Minimizing the squared numerator and denominator terms leads to the method title, ***Method of Least Squares*** (15:16–17).

Least squares works by introducing a reference trajectory (orbit) that is "close" to the estimated trajectory. An initial orbit determination method or a previous estimation run creates the reference. The observation data is then used to compute residuals (observed value minus predicted value) which correct the initial trajectory in an iterative fashion. Further iterations are calculated if

the first correction does not eliminate all trends in the residuals. After convergence, the estimation process is complete and the new state vector completely describes the orbital trajectory (15:68).

2.4 Matrix Background

2.4.1 The State Transition Matrix. Systems with non-linear equations of motion may be described as

$$\dot{x} = f(x,t) \tag{4}$$

where x represents the reference state vector. By linearizing these equations through the use of a Taylor series expansion, a solution can be obtained in the following form:

$$\delta\dot{x} = \left.\frac{\partial f}{\partial x}\right|_{x_o} \delta x \tag{5}$$

The linearization is about a reference trajectory retaining only the first order terms. This results in a homogeneous equation with a fundamental solution called the state transition matrix, Φ. The solution to the *equations of variation* in Equation 5 can be written as

$$\delta x(t) = \Phi(t,t_o)\delta x(t_o) \tag{6}$$

The Φ matrix possesses the following properties:

$$\begin{aligned} \Phi(t_o,t_o) &= I \\ \Phi(t_2,t_o) &= \Phi(t_2,t_1)\Phi(t_1,t_o) \\ \Phi(t_o,t_1) &= \Phi^{-1}(t_1,t_o) \end{aligned}$$

The state transition matrix is found either through numerical methods by finding $A = \partial f/\partial x$ with $\dot{\Phi} = A\Phi$, or in closed form when the solution to the dynamical system is known (as in the two-body

problem). The Φ matrix in the simplified least squares case for this study is solved in closed form in Appendix C (7:25–26)(14:114–116).

2.4.2 The Data Linearization Matrix. The nonlinear relationship between the observations, z, and the instantaneous orbital state vector, x, can also be linearized through a Taylor series expansion. The equations can be written in matrix form as

$$\delta z(t) \approx H \delta x(t) \tag{7}$$

where H holds the information about this linear relationship. H is the matrix of partial derivatives of the observations with respect to the instantaneous orbital state and are evaluated at each observation time. The data linearization matrix used in this study is also solved in closed form and is found in Appendix D (7:26)(15:65).

2.4.3 The Covariance Matrix. At any point in the estimation process, the current reference trajectory describes the best state of the system. The best state is subject to a performance measure contained within the symmetric covariance matrix, P:

$$P = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1N}^2 \\ \sigma_{12}^2 & \sigma_{22}^2 & \cdots & \sigma_{2N}^2 \\ \vdots & \vdots & & \vdots \\ \sigma_{1N}^2 & \sigma_{2N}^2 & \cdots & \sigma_{NN}^2 \end{pmatrix} \tag{8}$$

The estimates of the components of the state vector are not definitively known but can be quantified with the individual elements of the covariance matrix. Random values of the state vector can have an effect on each of the elements and this is described as correlation between the data. The diagonal terms, σ_{ii}^2, are called variances and the off-diagonal terms, σ_{ij}^2, are covariances. Non-zero covariance

terms describe the statistical dependence between the variables. Zeros in the off-diagonal terms represent statistical independence of the variables (7:27)(15:22).

There are two covariance matrices associated with a dynamical solution to an orbit problem. The first is the dynamics covariance, P, and the second is the data covariance, Q. If the dynamics covariance is given at time t_o, the covariance, P, can be determined at any other time, t, with the state transition matrix, Φ:

$$P(t) = \Phi(t, t_o) P(t_o) \Phi^T(t, t_o) \tag{9}$$

The data covariance holds the statistical information about the accuracy of the individual observations. Each observation type has an associated standard deviation, σ (the square root of the variance), which takes care of any possible differences in units between the data types. The Q matrix has the individual variances positioned along the diagonal with zeros in the off-diagonal positions. This means that the individual observation types are assumed to be independent (7:27)(15:22,30,59–60).

2.5 Conclusions

While it appears that a sequential orbit estimator is the superior choice for routine updates, the batch filter has a suitable use for data reduction. Sequential filters are very useful for the constant update of location information, but for reducing an entire arc of observation data into an element set and a covariance matrix, a batch estimator is the proper choice for the initial examination. If the batch estimation techniques prove to be inadequate, then the sequential routines can be used for the initial estimation. Nonlinear least squares has a straightforward algorithm for processing a batch of data such as that associated with an arc of observations from a remote tracking station.

2.6 Summary

Orbit determination is a centuries-old process, refined through the years into a state-of-the-art method for determining the motion of orbiting objects. Batch or sequential data processing provides a broad classification of techniques. Batch mode estimators work with large blocks of data at one time and must have provisions for observation storage. This lends itself to analysis of trends within the data and can be performed at any time after the data has arrived. Sequential processing handles individual observations in real time. This means data storage requirements are much less than with batch modes and the internal matrix inversions are less complicated. The main disadvantage of sequential filters is detecting local trends in the data that might cause the routine to diverge.

The most common batch mode estimator, called the Method of Least Squares, was developed by Gauss in the late eighteenth century and is based on probability theory and the optimistic hope that a reference trajectory is close to the actual trajectory. This gives a solution to the state of a satellite that is the best possible as described by the Central Limit Theorem. Even though the data contain errors, the linearized dynamics are a good enough mathematical model to iteratively determine the orbital path.

There are several matrices associated with the differential correction process and they are classified as linearization matrices and covariance matrices. The linearization matrices include the dynamics linearization called the state transition matrix, Φ, and the data linearization, H, and they allow for an approximate solution to a non-linear dynamical system. The dynamics and the data each have a corresponding covariance matrix. The dynamics covariance matrix is P, which is symmetric, and the data covariance matrix, Q, is diagonal. Iterative calculation with each of these matrices results in a correction to the reference state which is declared the new estimate when the process has converged.

III. Methodology

3.1 Introduction

The programming involved for this study involves two main parts: the truth model and the differential corrector. The truth model generates observation values that represent an actual orbit of an Earth satellite. The model outputs range, azimuth, and elevation points for several tracking stations at a specified time interval. Building the model involves using mathematics based on classical astrodynamic relationships. The differential corrector is much more math intensive. Through probability theory and matrix manipulation it corrects a reference trajectory based on the input observations from the truth model. The estimation routine iterates finding a solution after convergence. This chapter includes a detailed discussion of the construction of both the truth model and the differential corrector.

The accuracy of the theoretical methods used in the programs is checked through various methods. The truth model is checked against a Pascal program written by Dr. Kelso which verifies correctness in the observation values generated, as well as event times for each of several stations. Elements of the differential corrector are checked through both numerical and closed-form solution methods. Also, several routines are checked through hand calculation and by using different expressions for the calculations.

The actual research itself is conducted through a Monte Carlo analysis based on five orbit types. Individual runs are checked for a complete reduction of observations to element sets by examining the resultant root-mean-square values. Several test cases are examined to check for differences between one pass and several passes at the same site.

3.2 Justification for Method Selected

The use of the truth model to generate the data needed for the differential corrector (DC) is far superior to using actual data. Actual data contains unknown sigmas and biases that cannot be

accurately modeled. Any discrepancies in the data make the debugging of the DC routine nearly impossible. Using the truth model data eliminates one cause for uncertainty. The truth model also lends itself to generating data of varying degrees of accuracy by including different perturbation effects. At first, two-body data alone is used to check the accuracy of the differential correction routine, and after validation with this data, more realistic data can be used for the actual analysis.

Use of the differential corrector with two-body dynamics alone simplifies the process of orbital estimation. Estimating an orbit based on a short track of data does not lend itself to determining all perturbing effects on the orbit. The simplification of using only two-body dynamics allows for the closed-form solution of several matrices involved in the correction process. This means accuracy is only limited by the precision of the machine running the program. Least squares is the easiest dynamical model to initially program. All other estimators rely on the same mathematical relationships as least squares but handle various elements in the correction process in a different fashion. Once the least squares programs are working according to expectations, the other routines, such as sequential estimators, can be explored.

3.3 Theoretical Background

This section is provided as a description of the FORTRAN programs used in the study. File formats and program methodology are outlined to supplement the sometimes brief comments within the programs themselves. The code assumes an understanding of the math and iteration processes and comments are included only as specific identifiers to the mathematical formulas and intricacies of FORTRAN. The following sections provide detailed theory development of the equations used both in the truth model and in the differential corrector.

3.3.1 The Truth Model. The truth model is a data-creating program that generates satellite positional information when given as an input an initial state vector, epoch time, and integration step size. The use of a truth model helps eliminate programming errors in the differential corrector

by supplying "perfect" data to the corrector routine. Actual data from the sites are subject to errors and biases while truth model data can be controlled and regulated. The truth model data is based on a two-body orbit with perturbations added and outputs range, azimuth, and elevation observations for a set of remote sensor locations. In order to simulate actual tracking observations, a Gaussian random number generator introduces noise to the data based on user input observation sigmas.

The input file to the model, labeled 'input.t,' contains the start and stop time specified by year, month, day, hour, minute, and second (reference the program listing in Appendix E). Next, the file contains the integration step size in seconds. The last item in the file is the reference state vector for the desired orbit specified as a position and velocity in kilometers and kilometers/second. The time, as input, is not adequate for calculation purposes. The subroutine 'julday.for' converts the input format to a Julian day (minus 2440000.0) which is then converted to seconds for the numerical integration routine.

The numerical integration routine is a fourth-order predictor-corrector method called Hamming. It is an ordinary differential equations integrator which carries the last four values of the state vector which are extrapolated to obtain the next, or predicted, value along the polynomial fit trajectory. After the the predictor part, the extrapolated data point is corrected based on a high-order polynomial to obtain the new value for the state vector. Initially, the routine is supplied with a single state vector from which three other values are obtained through a Picard iteration (by polynomial extrapolation). Once the routine is up and running, four values are maintained throughout the integration interval and the Picard iteration is bypassed.

While Hamming is the central looping routine for propagating a state vector to generate ephemerides, a subroutine call is made within 'haming.for' requesting the evaluation of the equations of motion (EOM). The equations are handled in the routine called 'rhs' for the right-hand side of first-order differential equations. The dynamical terms are based on two-body orbital motion given

by

$$r = \sqrt{x^2 + y^2 + z^2} \tag{10}$$

$$\dot{x} = v_x \tag{11}$$

$$\dot{y} = v_y \tag{12}$$

$$\dot{z} = v_z \tag{13}$$

$$\dot{v}_x = -\frac{\mu x}{r^3} \tag{14}$$

$$\dot{v}_y = -\frac{\mu y}{r^3} \tag{15}$$

$$\dot{v}_z = -\frac{\mu z}{r^3} \tag{16}$$

where x, y, and z are the three values for the position of the satellite and v_x, v_y, and v_z are the three values for the velocity of the satellite.

The next dimension of creating a truth model that closely resembles reality is to introduce the dominant perturbation effects on the acceleration terms $\{\dot{v}_x, \dot{v}_y, \dot{v}_z\}$ above. The two largest perturbations for lower-orbiting earth satellites are due to the J_2 term of the earth's zonal harmonic, and air drag. To include J_2, the gradient of the potential term must be subtracted from the existing two-body acceleration solution. The infinite series expansion of the geopotential in spherical harmonics is

$$V(r, \theta, \phi) = -\frac{\mu}{r} \sum_{n=0}^{\infty} \sum_{m=0}^{n} \left(\frac{r}{R_e}\right)^{-n} P_n^m(\cos\theta)(C_{nm} \cos m\phi + S_{nm} \sin m\phi) \tag{17}$$

where C_{nm} and S_{nm} specify the shape of the gravitational field. To extract the J_2 potential term, set $n = 2$ and $m = 0$ and making the substitution that $J_2 = -C_{20}$. Then,

$$V_{20} = \frac{\mu R_e^2 J_2}{2r^3}(3\cos^2\theta - 1) \tag{18}$$

with $J_2 = 0.0010827$ and $R_e = 6378.137$ km (14:55–59).

Now, to convert this potential equation into a useful form, it must be converted into x, y, and z coordinates and then the partial derivatives are subtracted from the two-body solution. From Figure 2,

$$\cos(90 - \theta) = \sin\theta = \sqrt{\frac{x^2 + y^2}{x^2 + y^2 + z^2}}$$

and

$$\cos^2\theta = 1 - \frac{x^2 + y^2}{x^2 + y^2 + z^2}$$

which is substituted directly giving

$$V_{20} = \frac{\mu R_e^2 J_2(-x^2 - y^2 + 2z^2)}{2(x^2 + y^2 + z^2)^{5/2}} \tag{19}$$

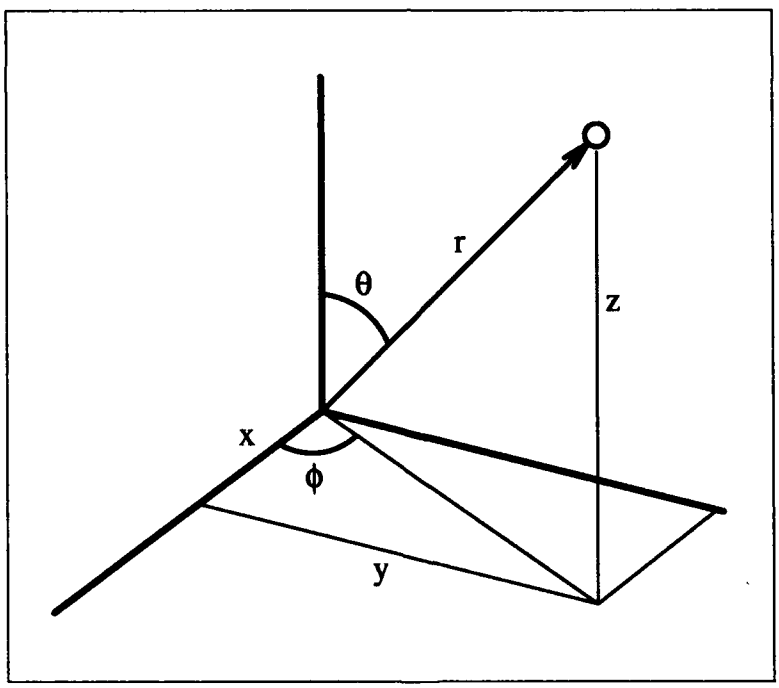


Figure 2. Spherical Coordinates for Geopotential Harmonics

The partial derivatives are derived from ∇V_{20}:

$$\frac{\partial V_{20}}{\partial x} = \frac{\mu x}{r^3}\left[\frac{3}{2}J_2\left(\frac{R_e}{r}\right)^2\left(1-\frac{5z^2}{r^2}\right)\right] \tag{20}$$

$$\frac{\partial V_{20}}{\partial y} = \frac{y}{x}\frac{\partial V_{20}}{\partial x} \tag{21}$$

$$\frac{\partial V_{20}}{\partial z} = \frac{\mu z}{r^3}\left[\frac{3}{2}J_2\left(\frac{R_e}{r}\right)^2\left(3-\frac{5z^2}{r^2}\right)\right] \tag{22}$$

The μ/r^3 terms out front are pulled out because they are identical to the two-body terms and make programming easier.

Since this study will include lower earth-orbiting satellites, accelerations due to air drag are also subtracted from the the two-body terms. Determining air drag through the changing atmosphere is not exact, but the form of the acceleration is

$$a_d = -\frac{1}{2}B^*\rho v\vec{v} \tag{23}$$

where

$$B^* = \frac{C_D A}{m}$$

with $C_D = 2$, $A = 7.5\text{m}^2$, and $m = 1000$ kg (used as representative values for calculation purposes) (14:65). The density of the atmosphere, ρ, is empirically determined and currently governed by the 1976 Standard Atmosphere (see Figure 3).

To determine the drag value, an appropriate density must be determined for the satellite's orbit altitude:

$$\rho(h) = \exp\left[-\left(\frac{z_1}{h}\right)^4+\left(\frac{z_2}{h}\right)^3-\left(\frac{z_3}{h}\right)^2+\left(\frac{z_4}{h}\right)-\left(\frac{h}{z_9}\right)^4+\left(\frac{h}{z_8}\right)^3-\left(\frac{h}{z_7}\right)^2+\left(\frac{h}{z_9}\right)-z_5\right]$$

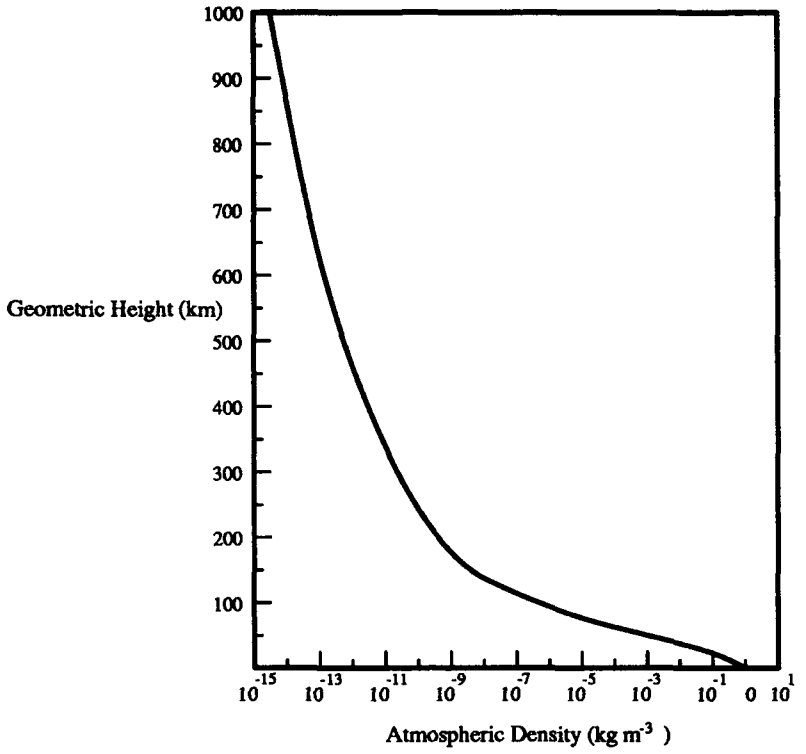


Figure 3. Density versus Geometric Height for the 1976 U.S. Standard Atmosphere (13:62)

where

$$z_1 = 246482.873 \qquad z_4 = 19174788.0 \qquad z_7 = 56060.0780$$

$$z_2 = 537132.030 \qquad z_5 = 105.381650 \qquad z_8 = 160314.760$$

$$z_3 = 1536228.60 \qquad z_6 = 4968.56120 \qquad z_9 = 344944.780$$

and h is the altitude. This density model, as is the 1976 standard model, is valid for altitudes of 100–1000 km (3).

The next term, $v = |\vec{v}|$, is the magnitude of the velocity of the vehicle relative to the atmosphere. The rotating atmosphere velocity vector, $\vec{v}$ is defined as

$$\vec{v} = \begin{bmatrix} v_x + \omega_e y \\ v_y - \omega_e x \\ v_z \end{bmatrix} \tag{24}$$

where ω_e is the rotation rate of the earth (1:423–424).

The air drag equation appears as a good deterministic model, however, the B^* and ρ terms can change drastically. B^* depends on the rotation and orientation of the vehicle as it orbits the earth and is highly time dependent. The density model also changes with time and is affected by local wind variations, surface heating, the equatorial bulge, and solar flares (14:65).

With the equations of motion information available, the Hamming routine supplies a new position and velocity data point at each step-size increment. These state vectors need to be converted into observations of range, azimuth, and elevation which are site dependent. The sites used are the Air Force Satellite Control Network (AFSCN) remote tracking stations and are listed by name and location in Table 1.

Table 1. Remote Tracking Stations of the AFSCN

Station	Latitude (deg)	Longitude (deg)	Altitude (m)
Indi	-4.671747860	55.477820590	560.500
Reef	-7.270030560	72.369998600	-68.375
Guam	13.615187820	144.856049380	218.930
Hula	21.562265240	201.757894060	429.420
Cook	34.822598900	239.498147050	271.530
Pike	38.805943055	255.471532222	1899.420
Boss	42.947821440	288.373437430	203.370
Pogo	76.515364390	291.401141690	147.030
Lion	51.117583380	359.093654500	146.590

The conversion from ECI to topocentric elements is discussed in detail in Section B.5 of Appendix B. Each site in 'sensors.loc' has a separate output file that will be written to if there is visibility (elevation ≥ 0).

The truth model data generates "perfect" data for use in the differential correction routine. While this is an excellent source of data to debug any errors in the estimation process, it is not a very good representation of reality. To simulate errors, or "noise", a Gaussian random number is generated and multiplied by the sigma associated with the particular observation type. The

Gaussian random number generator outputs a value that is distributed according the the curve in Figure 1. This means that 99 percent of the time the number generated will be between -3.0 and 3.0 (see Figure 5). Each call to the observation calculation routine has a different seed value for the random number generator which will produce a new random multiplier. The random number, multiplied by the appropriate sigma value, is added to the observation. This process distributes the errors according to the Central Limit Theorem as discussed in Chapter 2 and is typical for any instrument.

3.3.2 The Differential Corrector. The least squares estimator, 'leastsq.for,' is a matrix intensive looping routine to process observations and obtain an estimate of the satellite state vector. Observations are handled together as a batch. An equinoctial element set is maintained as the reference orbit which must be "close" to the orbit contained within the data. For each data point, a predicted observation set of range, azimuth, and elevation is generated from the equinoctial state equations of motion and is compared to the actual observation to get a residual value. By summing up the correction terms for each calculated residual, the reference trajectory is adjusted. At the same time, a covariance matrix is created in the same summing fashion for each observation. The covariance matrix contains the confidence level for the estimation process. After the correction, if the new equinoctial state is within tolerances, the process has converged upon a solution, but if the correction is relatively large, the process is repeated for another iteration, giving another correction to the reference state. The process either converges to a solution within the maximum number of allowed iterations, diverges, or reaches the maximum iteration value without a definite solution.

The input file for the least squares routine, 'input.d,' controls the iteration process and supplies the data (reference the program listing in Appendix F). The first entries are the epoch time in year, month, day, hour, minute, and second which is converted to a Julian day format in the subroutine 'julday.for,' and the reference state in kilometers and kilometers/second which is converted into equinoctial elements in the subroutine 'equin.for.' Next, the maximum number of

iterations is included along with the residual rejection criteria entered as a one-sigma multiplying factor. The last entry in the input file is the data itself, with the date entered as above, the station identifier associated with the observation point, and the observation data of range in kilometers, and azimuth and elevation in degrees. The observations contained in the input file can be for a single track of data or multiple tracks as long as the station identifier is maintained for each point.

To reduce the amount of error associated with propagating the equations of motion of the equinoctial elements from the reference time to each observation time, the same Hamming routine from the truth model is used in the differential corrector before the data is processed. The integrator works exactly the same as before including two-body effects along with J_2 and air drag perturbations. The new reference point derived has an epoch time equal to the time of the first observation. Because 'haming.for' works with units of kilometers, kilometers/second, and seconds, the reference state is not converted to equinoctial elements (see Appendix B) and canonical units until after it has been integrated forward.

Least squares is an iterative process. The iteration loop handles the program overhead of reinitializing matrices, resetting counters, updating the reference trajectory, and checking for convergence. Within each iteration, all of the observation data is processed including routines to propagate the state vector, obtain the linearization matrices, and to create residuals, the state correction, and the covariance matrix. A flowchart outlining the least squares algorithm is shown in Figure 4.

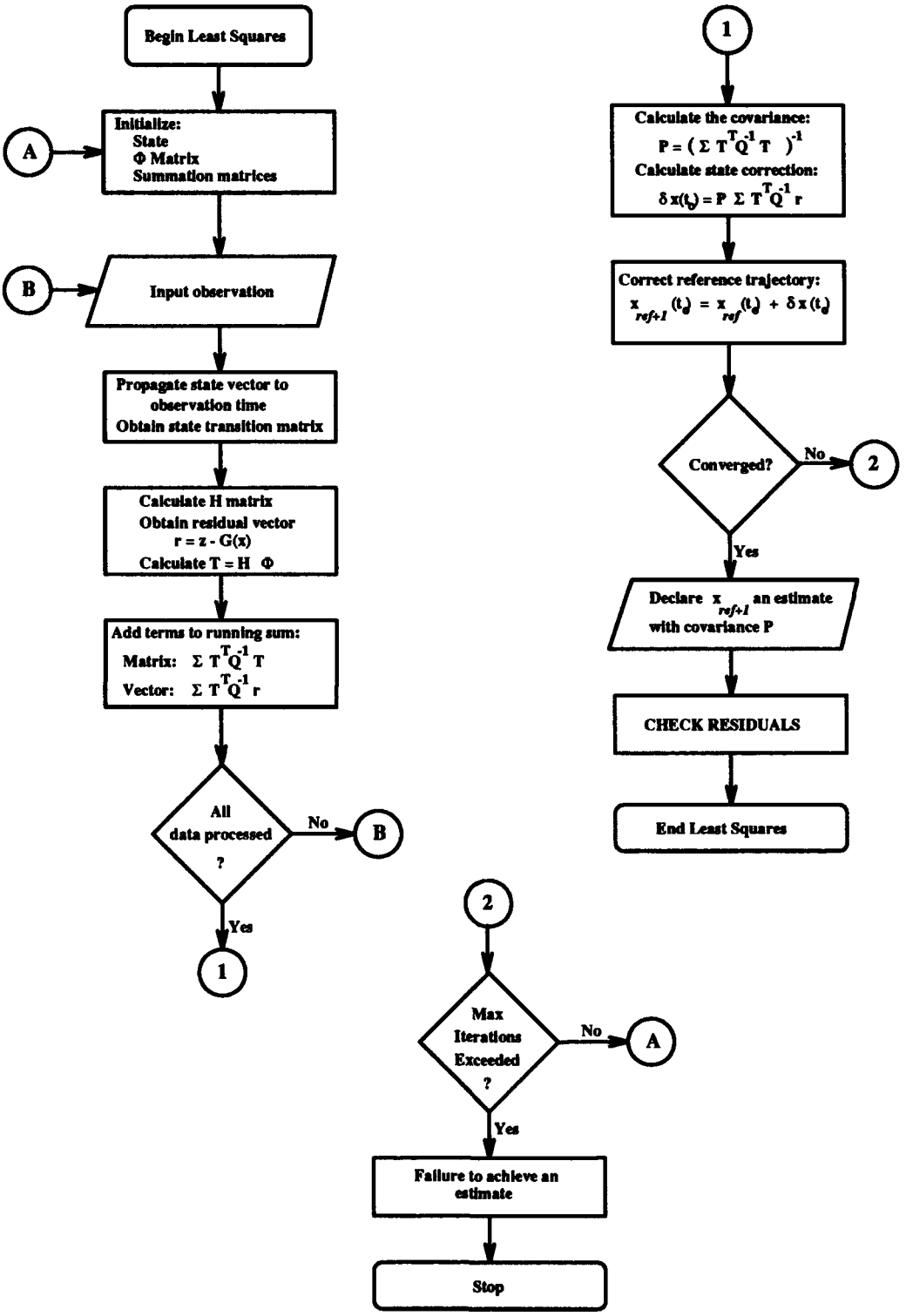


Figure 4. Non-Linear Least Squares Algorithm

The observation processing loop controls the matrix summing routines needed to update the satellite state. The matrices are built by accumulating residual values from single observation sets. Each range, azimuth, and elevation observation contains information about the actual state of the satellite and residuals are formed by subtracting the predicted observations. The actual observations are contained in the input file and processed sequentially. The predicted observations are based on the reference equinoctial element set, propagated forward with the equations of motion described in Appendix C.

Section 2.3 outlines the basic mathematical foundation for least squares. Recall the minimization problem of Equation 3:

$$\frac{d}{d\bar{x}} \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{2\sigma_i^2} = 0$$

For N independent observations, the probability density can be written as

$$P(x_i) = (2\pi)^{-N/2} \left[\prod_{i=1}^{N} \sigma_i^{-1} \right] \exp\left(- \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{2\sigma_i^2} \right) \tag{25}$$

and then in matrix form as

$$f(x) = (2\pi)^{-N/2} |Q|^{-1/2} \exp\left[-\frac{1}{2} (x - x_o)^T Q^{-1} (x - x_0) \right] \tag{26}$$

with the data covariance matrix, Q defined as

$$Q = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{pmatrix} \tag{27}$$

The zero off-diagonal elements in the Q-matrix represent the statistical independence of each element in the data vector, z_i, and N is the number of observations used in the estimate (15:17,22).

With the state transition matrix available (see Appendix C), it is possible to write the linearized equations of motion as

$$\delta x(t) = \Phi(t, t_o)\delta x(t_o) \tag{28}$$

hoping that the correction, $\delta x(t)$, is small. The linearization of the dynamics occurs about the reference trajectory contained in the initial element set (which was converted to equinoctial elements). In order to form the residual vector, the observation relation $G(x_{ref}(t_i), t_i)$ must also be linearized. This data linearization, $H = \partial G/\partial X$, is solvable in closed form for the equinoctial element problem and is discussed in detail in Appendix D (15:65). The program that handles the computation of the H matrix is 'obser.for.'

The matrix handling 'obser.for' controls three key elements for the least squares processing. First, the data covariance elements are assigned to the diagonal elements of the Q matrix. Typical sigma values for AFSCN tracking stations are 100 meters in range, and 0.025 degrees in azimuth and elevation. To meet the format requirements of Q, each sigma is squared and inverted after being converted to the proper canonical units. The second function in this subroutine is to calculate the predicted observation vector of range, azimuth, and elevation from the input equinoctial element set. This transformation is outlined in detail in Appendix B. The last routine is the aforementioned calculation of the data linearization matrix H. The entirety of Appendix D is devoted to the development of this matrix.

Least squares estimation assumes that the observation data at time t_i is linearly related to the state at the same time. By accepting the residual as an approximation to the true error, the actual data is expressed as

$$z_i(t_i) = G(x(t_i), t_i) \tag{29}$$

Now, the expression for r is

$$\begin{aligned} r_i \approx H_i \delta x(t_i) &= H_i \Phi(t_i, t_o) \delta x(t_o) \\ &= T_i \delta x(t_o) \end{aligned} \tag{30}$$

where $\delta x(t_o)$ is the correction to the reference state. Substituting this into the exponent argument of Equation 26 leads to a new minimization problem of

$$\frac{\partial}{\partial \delta x} \left[(z - T\delta x)^T Q^{-1} (z - T\delta x) \right] = 0 \tag{31}$$

Solving this equation leads to the results

$$\delta x(t_o) = (T^T Q^{-1} T)^{-1} T^T Q^{-1} r \tag{32}$$

$$P_{\delta x} = (T^T Q^{-1} T)^{-1} \tag{33}$$

and the desired estimate is

$$\bar{x}(t_o) = x_{ref}(t_o) + \delta x(t_o) \tag{34}$$

if the estimation process has converged to a solution (15:57–66).

The form of the solution is not efficient due to the large matrix computations. To simplify the burden, an equivalent form is used when programming:

$$P_{\delta x} = \left(\sum_{i=1}^{N} T_i^T Q_i^{-1} T_i \right)^{-1} \tag{35}$$

$$\delta x(t_o) = P_{\delta x} \sum_{i=1}^{N} T_i^T Q_i^{-1} r_i \tag{36}$$

This form lends itself to the iterative process mentioned throughout this study. Individual observations are processed one by one and the matrix sums accumulate the correction and covariance values (15:60).

Along with each data point, a residual is calculated representing the error between the actual observation and the predicted value. These residuals are printed out after the first and last pass to give a second indication that the estimation process converges upon a good solution. The covariance matrix, P, contains the confidence level of the estimate, but it alone cannot be trusted because it does not depend on the residuals. The actual residual values must also be small.

Small residuals is a relative concept. The actual accuracy of the measurements is contained in the data covariance matrix, Q. The test on whether a computed residual should be added to the running sum of the correction and covariance matrix is done with each observation. The rejection criteria is contained in the input file as a sigma multiplier and this, combined with the sigma values in the Q matrix, specifies the rejection threshold. If the computed residual is higher than the threshold, the observation point (range, azimuth, and elevation) is rejected. Because an observation was rejected on one iteration should not exclude it from further consideration. If a correction is made to the state, and the least squares routine has not yet converged, all data is re-processed with new residuals computed and checked for rejection. This repetitive process keeps the estimation routine from completely discarding potentially good data that caused large outliers on an early iteration.

The diagonal elements of the dynamics covariance matrix contain the accuracy of the estimate produced and they also hold the key for convergence. The six standard deviations, σ_{ii}, form a six-dimensional error ellipsoid describing the most probable location of the satellite. When the resulting correction is within a small fraction (such as 1/100th in this program) of the one-sigma ellipsoid, there is no need to continue updating the state. If convergence is not obtained, another iteration is performed until the reaching maximum iteration number of 15.

After convergence another check based on residual values is performed. The root-mean-square values for each observation type is calculated by summing the squares of the residuals divided by the number of data points and taking the square root of the division result. The data covariance holds information on the accuracy of the data (15:67-68). If the estimation routine is doing an adequate job, the RMS value should be about equal to the sigmas contained in the Q matrix. If the RMS values are larger than the sigmas, the process has not removed all of the information contained in the data resulting in a sub-standard estimate of the orbit.

Once least squares has converged, the new estimate with covariance are written to the output file 'output.d.' The estimate is made by correcting the reference state as described in Equation 34. The inverse of the covariance is accumulated throughout the least squares iterations and an IMSL routine for symmetric matrices solves for P. The IMSL routine provides for an iterative refinement on the solution to come up with the most accurate inverse available. This two-step routine to determine an inverse helps avoid problems due to poorly conditioned matrices (widely varying exponent values) and singularity problems.

The last function of the differential corrector is to output the covariance matrix, P, at the time of the last observation. This is a useful step, particularly if the new estimate is going to be used in a follow-on estimator. The state transition matrix provides the means for propagating the P matrix and is specified in Equation 9 and handled by the subroutine 'phipphi.for.'

Divergence in the routine occurs when the least squares solution systematically adds larger and larger corrections to the reference trajectory until it appears as a maximized solution rather than a minimized solution. Divergence can also occur when corrections are added to the state on one iteration and then subtracted on the next iteration. The pendulous motion makes it difficult for the estimation routine to find an exact solution. The k and h elements are especially sensitive to the size of the correction and can cause eccentricity values to quickly become greater than one.

3.4 Research Methodology

The truth model data is based on two-body orbital motion perturbed by J_2 and air drag. Simulated noise alters the data by adding the product of a Gaussian random number multiplied by standard deviation values for each observation type. Five separate truth model runs generate the data for the desired sample orbits and store the range, azimuth, and elevation into separate station files as individual passes. Elevations greater than zero indicate visibility for a particular station.

The five sample orbits have unique characteristics to expose any possible deficiencies in the equinoctial element differential corrector. Good performance of the corrector is desirable for the low altitude orbits because a majority of earth-orbiting objects are in low orbits. Testing begins with the high orbits, however, because the higher orbits have fewer perturbations and longer pass lengths. The highest orbit tested is for a Global Positioning System (GPS) satellite with an altitude over 20,000 kilometers. The next orbit comes from the high eccentricity/critical inclination Cosmos 1305 rocket body followed by the third orbit investigated from Explorer debris in a high inclination track. The two low-orbit satellites tested are the sun-synchronous Defense Meteorological Satellite Program (DMSP) at 850 kilometers altitude and the Russian space station, Mir, at 350 kilometers altitude.

Twenty Monte Carlo runs for each orbit with the differential corrector eliminate any one-time biases in the truth model data. The same pass is repeatedly examined based on twenty different sets of observations based on different seed values for the random number generator. A random pass is selected from the observation sets to show a close-up view of individual residual values. Overall performance is verified by comparing the resulting root-mean-square (RMS) values for each run with the desired one sigma level. If the trend of the twenty RMS values is near the one-sigma error, then the new estimate is a sufficient representation of the orbit contained in the original data.

3.5 *Equipment*

The FORTRAN programming language is used in the coding of all routines for this analysis. The only special requirement to run this code, is to have access to the IMSL library containing the matrix factoring and solving routines. All other facets of this analysis are not specific to any system and all results could be reproduced by running the programs on any system.

3.6 *Validation of Method*

The truth model data is generated from astrodynamic routines found in Reference (1) and Reference (8). The output data is verified by comparing it to Pascal routines written and developed by Dr. Kelso to compare visibility times at similar stations. Each orbit is tested against the Pascal version with differences of less than 10 meters for range, differences of less than 1/10th of a degree for azimuth, and 1/2 of a degree for elevation. The main elevation discrepancy comes from an atmospheric refraction correction factor included in the Pascal program but not in the truth model.

The math-intensive data linearization matrix, H, can be validated using numerical derivative techniques. H is solved in closed form and constructed through matrix multiplication. To check the numbers produced in the resultant 3×6 matrix, the closed-form $H = \partial G / \partial X_{equin}$ is computed numerically as

$$H = \frac{\Delta G}{\Delta X_{equin}} \tag{37}$$

Each of the equinoctial elements in X_{equin} is perturbed by a small amount (on the order of 10^{-5} to 10^{-7}). Subtracting the original $G = \{\rho, Az, El\}$ from the new perturbed value and dividing by the perturbation amount gives an approximation to the H values:

$$H = \frac{G_1 - G_o}{\Delta X} \tag{38}$$

As an example, to get the H_{11} element which represents $\partial\rho/\partial\ell$, ℓ is perturbed by 0.00001 and a new ρ value calculated. The old ρ value was 0.353920745 distance units, and the new ρ is 0.353911148 distance units giving

$$H_{11} = \frac{0.353911148 - 0.353920745}{0.00001} = -0.959700 \tag{39}$$

which is very close to the analytical calculation of H_{11} of -0.959707.

Verification of the equinoctial element equations of motion involved only the time varying element, $\ell = M$. Checking consecutive data points based on the mean motion verified that the element was being propagated correctly. The mean motion was calculated as $n = \sqrt{\mu/a^3}$ and when multiplied by the time increment between data points the answer should equal the equations of motion value.

The final check for an accurate estimator involves the covariance matrix and the RMS values. The covariance matrix should be symmetric and the product of P with P^{-1} should return a matrix very close to the identity matrix. The ultimate check on the differential correction process lies with the RMS values. If the computed RMS values for each observation are less than the associated sigmas for that observation type, then the correction process has exhumed all possible information out of the data and further correction steps are no longer needed.

3.7 Historical Accuracy of Method

Least squares as a differential correction routine has been around for centuries. This method of estimation was the first successful routine for estimating orbits of heavenly bodies through the solar system with a minimal amount of data available. This method has been used operationally for years at the Consolidated Space Operations Center (CSOC), the Consolidated Space Test Center (CSTC), and at the Space Surveillance Center (SSC). The equinoctial element set is also the preferred element set for earth-orbiting satellites to eliminate the difficulties discussed in Appendix A.

3.8 Statistical Procedures Review

The Gaussian random number generator is supposed to interject a dose of reality into the truth model data. Figure 5 shows the random distribution of the data for 500 data points. As shown in the figure, the data does appear random. The next verification step is to check if the random numbers generated are distributed normally as advertised. Figure 6 shows the distribution of the same 500 data points from Figure 5. These data points do appear to follow the Gaussian distribution as expected.

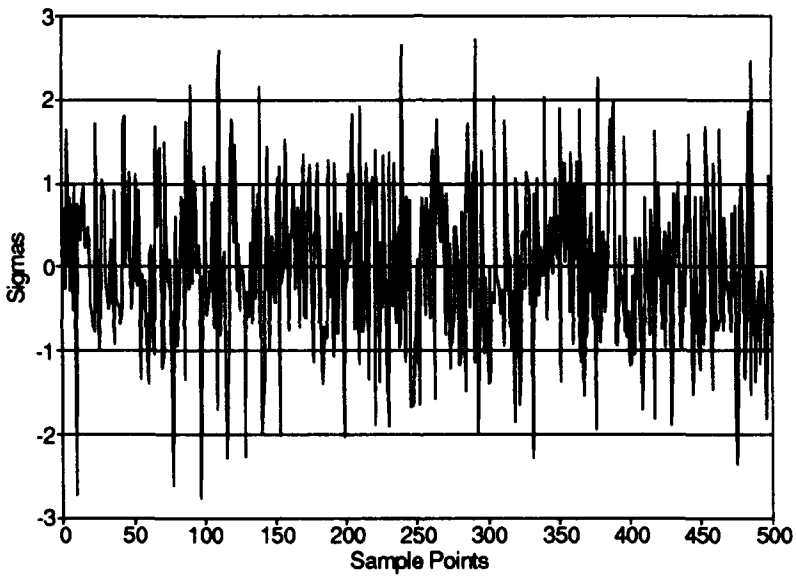


Figure 5. Verification of Random Number Generator

The performance of the differential corrector also shows differences when noise is added to the data. The residual values are indicators of how the estimation routine is performing. First pass residuals may be nowhere near the reference trajectory and it is the corrector's job to extract the information from the data and determine a new estimate of the state. The last pass residuals will then be within one standard deviation of zero. Figure 7 shows the residual values for each

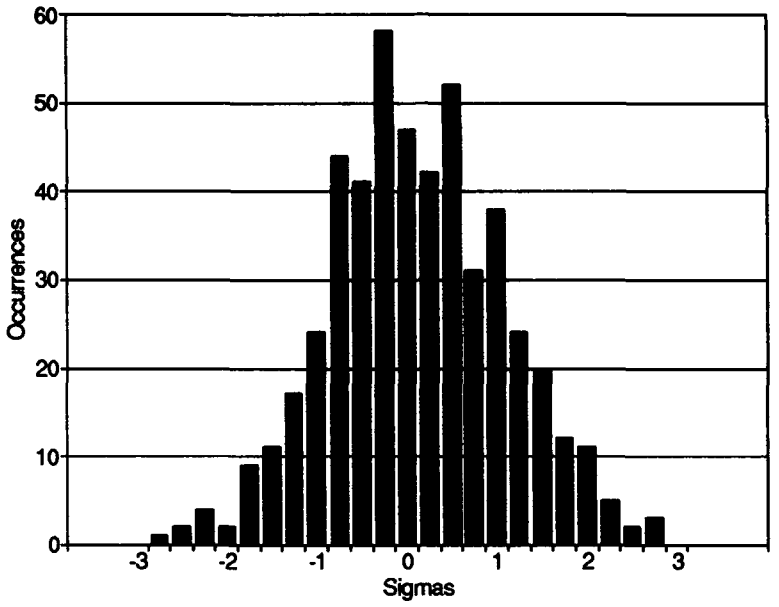


Figure 6. Histogram of Random Number Generation

observation type with no noise added to the data. Notice the much lower residuals after convergence, especially in the range values.

The Gaussian random number generator applies noise to the data but the performance of the differential corrector should still reduce the residuals to within one standard deviation. Notice in Figure 8 that the range residuals are pulled within the one-sigma value of 1.57×10^{-5} earth radii. Both the azimuth and elevation residuals before correction are already near the one-sigma value of 4.36×10^{-4} radians, so the corrector cannot do much more than mirror the data.

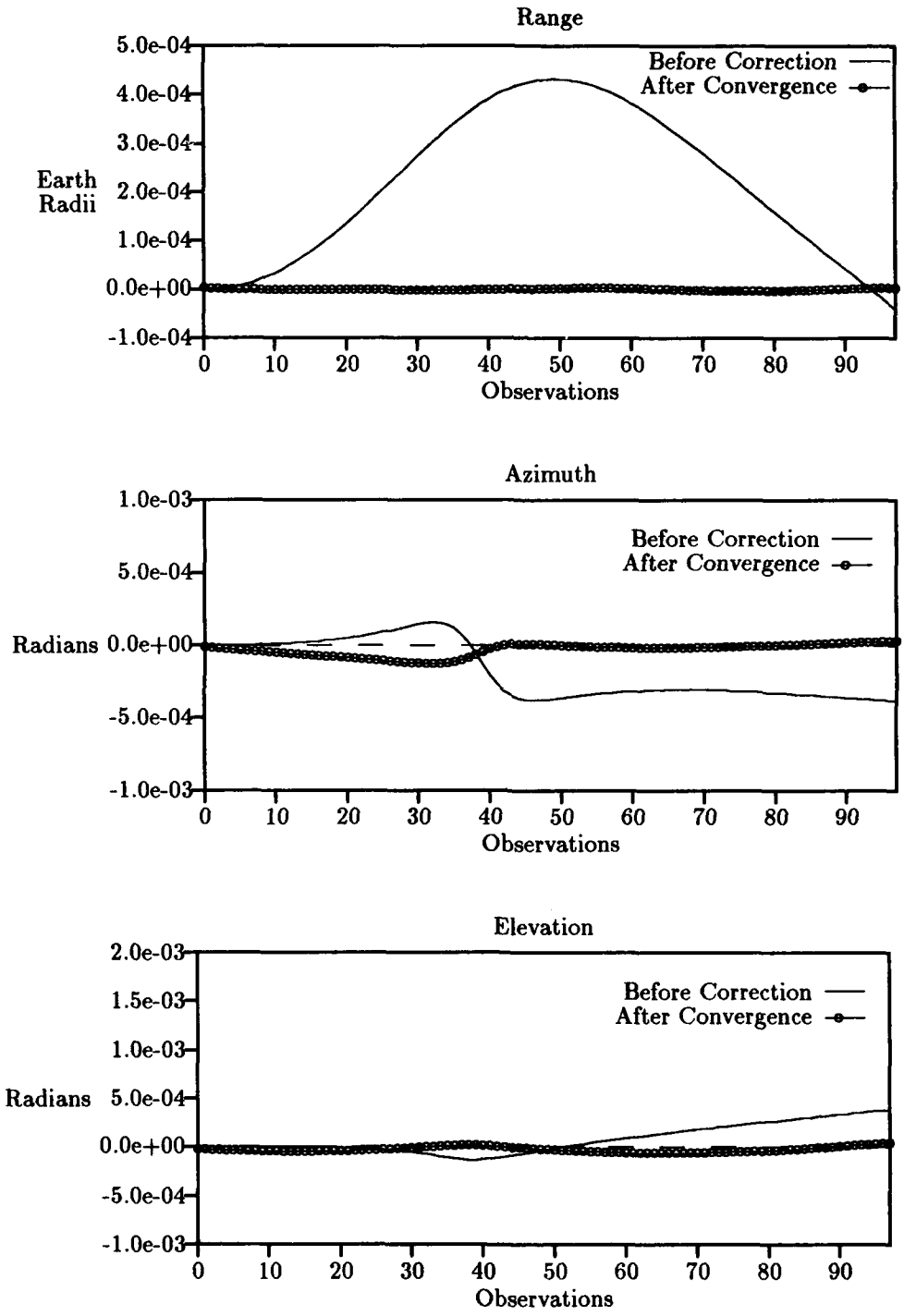


Figure 7. Residual Values with No Data Noise — GPS

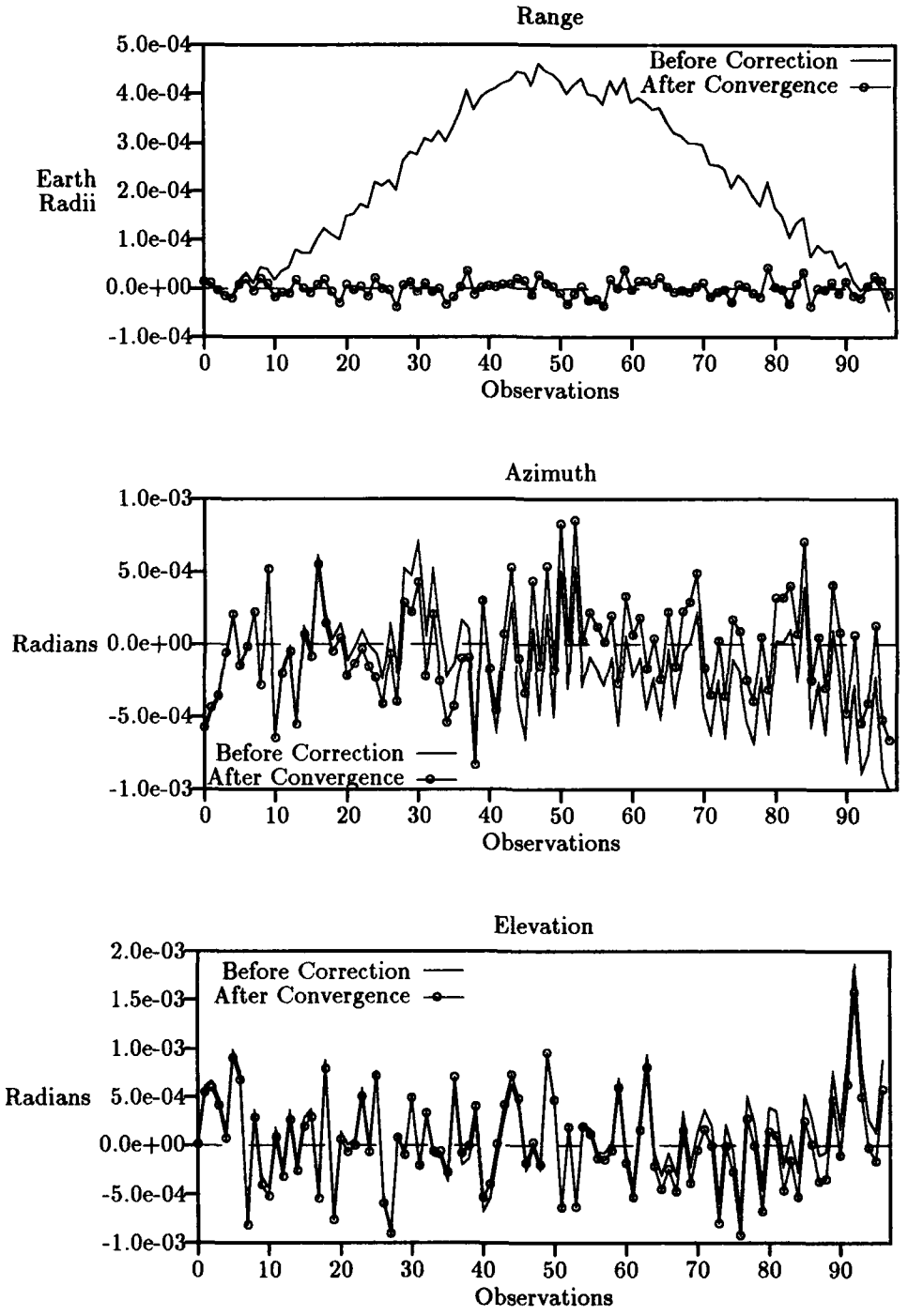


Figure 8. Residual Values with Data Noise — GPS

3.9 Summary

The mathematical development of the truth model is based on theoretical astrodynamic relationships as shown in Reference (1). The truth model produces ephemerides from an initial reference state vector which is propagated throughout an input time span using a fourth-order predictor-corrector integrator called Hamming. The differential corrector then tries to find a new estimate of the orbit through classical least squares batch methodology based on an equinoctial element reference vector and two-body equations of motion. Once the routine has completed, the root-mean-square values hold the accuracy information of the produced state vector.

IV. Analysis and Results

4.1 Introduction

Five different cases of satellite orbits are used in this study to determine the performance capabilities of the equinoctial differential corrector. The residuals from each estimation run are examined to ensure that high initial root-mean-square values are corrected to values near one sigma. Twenty individual Monte Carlo runs are also generated for each orbit to plot the overall trend of the RMS correction. This process is used to make sure the added random noise does not make it overly easy or difficult to converge for a single estimation.

4.2 Unit Considerations

Initially, the differential corrector performed all corrections in units of kilometers for distance and seconds for time (the same as in the truth model). Due to wide variations in covariance element magnitudes, there was difficulty inverting the 6×6 matrix to determine the covariance matrix, P. The two errors that kept re-occurring in the inversion routine were an ill-conditioned matrix and singular pivot entries or a singular matrix. In order to alleviate these difficulties, a unit change to canonical distance units (6378.137 km) and time units (806.8188744 sec) was performed. This solved most ill-conditioning problems but there were still singularity problems. The straightforward inversion routine provided by the IMSL library was not adequate. Since all covariance matrices are symmetric, a high-precision inversion algorithm solved the singularity problems. The first step is to factor the matrix and determine pivot points which are then input into a symmetric matrix solver routine. The covariance matrix is also improved through an iterative refinement procedure provided by IMSL. After these modificaticns to the model were made, there was no longer any problem with the inversion process.

4.3 Orbit Cases

The five test cases for the equinoctial differential correction routine are chosen due to unique orbital parameters in each case. The five orbits are listed in Table 2. Each case is fully examined using the twenty Monte Carlo run approach mentioned above.

Table 2. Orbit Cases for Differential Corrector Study

Case I	High Altitude, Low Eccentricity	GPS
Case II	High Eccentricity, Critical Inclination	Cosmos
Case III	Medium Altitude, High Inclination	Explorer
Case IV	Low Altitude, Sun Synchronous	DMSP
Case V	Low Altitude, Low Eccentricity	Mir

4.3.1 Case I — High-Altitude/Low-Eccentricity Orbit. The first orbit tested has a high altitude, therefore, it has relatively long pass lengths. The orbit used is from the Global Positioning System (GPS). The passes considered in this analysis were approximately eight hours in duration and used the remote tracking station site INDI. The advantages of using a high-altitude satellite orbit first is the long pass length means that a large percentage of the orbit is described by the observation data and there are minimal perturbation effects due to J_2 and air drag. Eccentricity is low and the inclination is not at any critical values. This is a stable and well known orbit and is useful for the initial analysis. The parameters of the GPS orbit are listed in Table 3.

Table 3. Orbit Elements — GPS

Cartesian			Classical			Equinoctial		
Epoch	09 Sep 92	10:12:00	Per	717.900	min			
x pos	-3031.911	km	a	26558.482	km	ℓ	4.097408	rad
y pos	-15025.844	km	e	0.006257		k	-0.004956	
z pos	21806.489	km	i	54.935	deg	ψ	-0.503224	
x vel	3.754356	km/sec	Ω	165.472	deg	L	2.040589	DU^2/TU
y vel	-0.889541	km/sec	ω	217.612	deg	h	-0.003819	
z vel	-0.114973	km/sec	M	234.764	deg	χ	0.130406	
			alt	20180.345	km			

The INDI pass examined was on 17 September 1992 from 05:05–13:05, approximately one week beyond the epoch time with data points spaced at five-minute increments. This results in

a reference trajectory that is one week old — presumably the maximum amount of time that a tracking station would use an estimate. Reference states are updated more frequently, but if the differential corrector is able to converge upon a solution with an old reference, then this validates the procedure with a more current reference. In Case I, a two-body-only integrator is used to propagate the reference trajectory from the epoch time to the time of the first observation. The perturbation effects are not needed to keep the reference "close." The rejection criteria is opened to a sigma multiplier of 3.0×10^5 to allow all observations to have an effect on the correction from the first iteration.

Figure 9 shows the overall corrector performance for the 98 data points in this pass. The goal is to take large residual values and reduce them below one sigma upon convergence. The overall view does show on a large scale that the residuals were reduced. In order to examine the relative values compared to one sigma, Figure 10 shows the final residual values after convergence in relation to the respective sigma values. While there are a few outliers, the overall trend in the data shows the corrector's success at extracting the orbital information from the original data.

The trends indicated in the previous figures are for a single data set from the truth model. To validate the effectiveness of the estimation process, the truth model provides twenty Monte Carlo data sets. (The previous figures show the results for Run 20). Figure 11 shows the overall performance of reducing the initial RMS values within one sigma. Figure 12 then shows the detailed view of the final RMS values. Note that there are final RMS values that are both above and below the one-sigma line but the overall average tends to follow the desired limit.

The twenty GPS runs show that the equinoctial differential corrector is effective for high-altitude orbits with minimal perturbation effects. Additional analysis for this orbit included reducing the number of data points to test whether the corrector could still converge to a solution. The number of data points was reduced finally to two values spaced 15 minutes apart and the corrector converged.

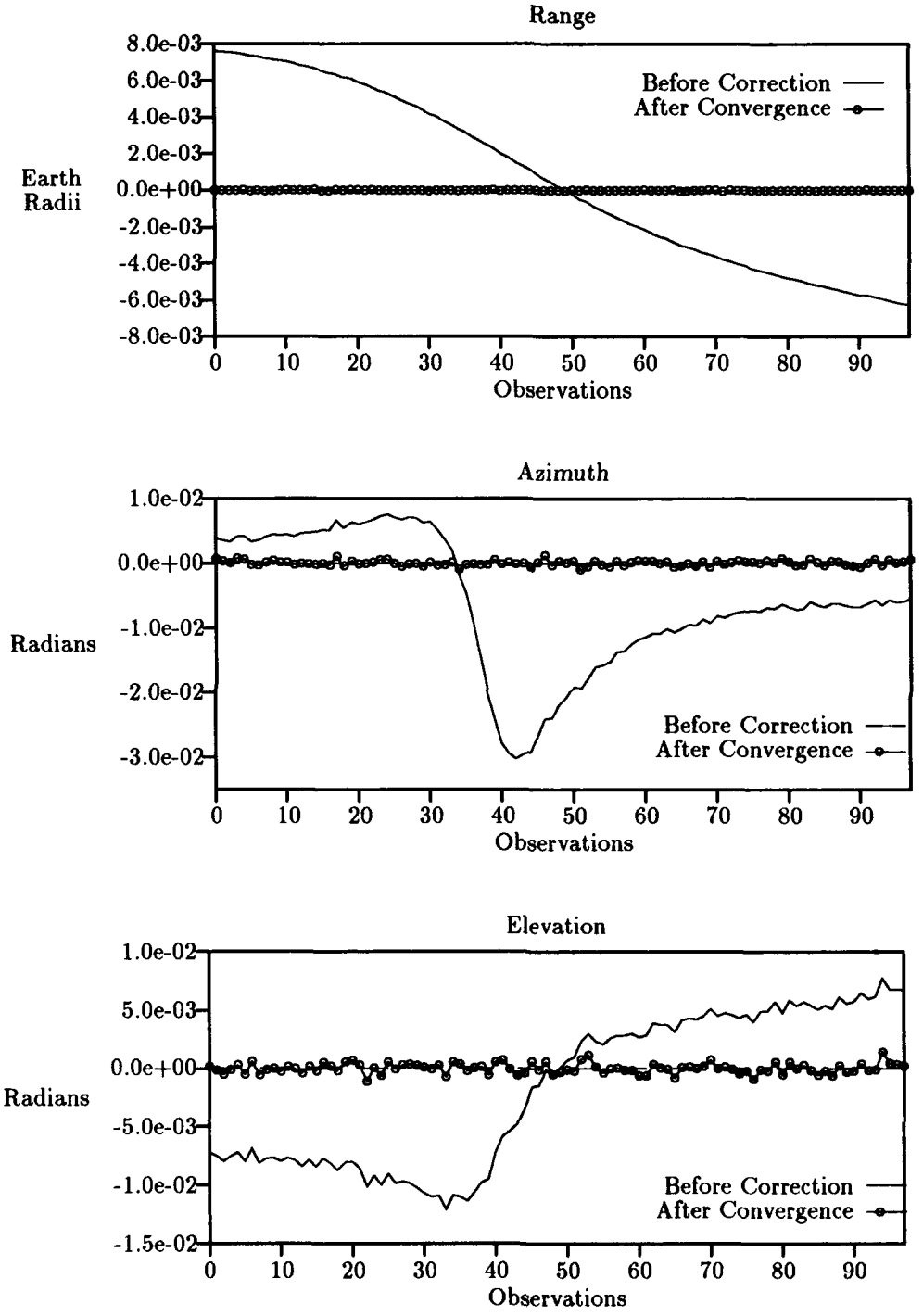


Figure 9. Residual Values Before and After Differential Correction — GPS

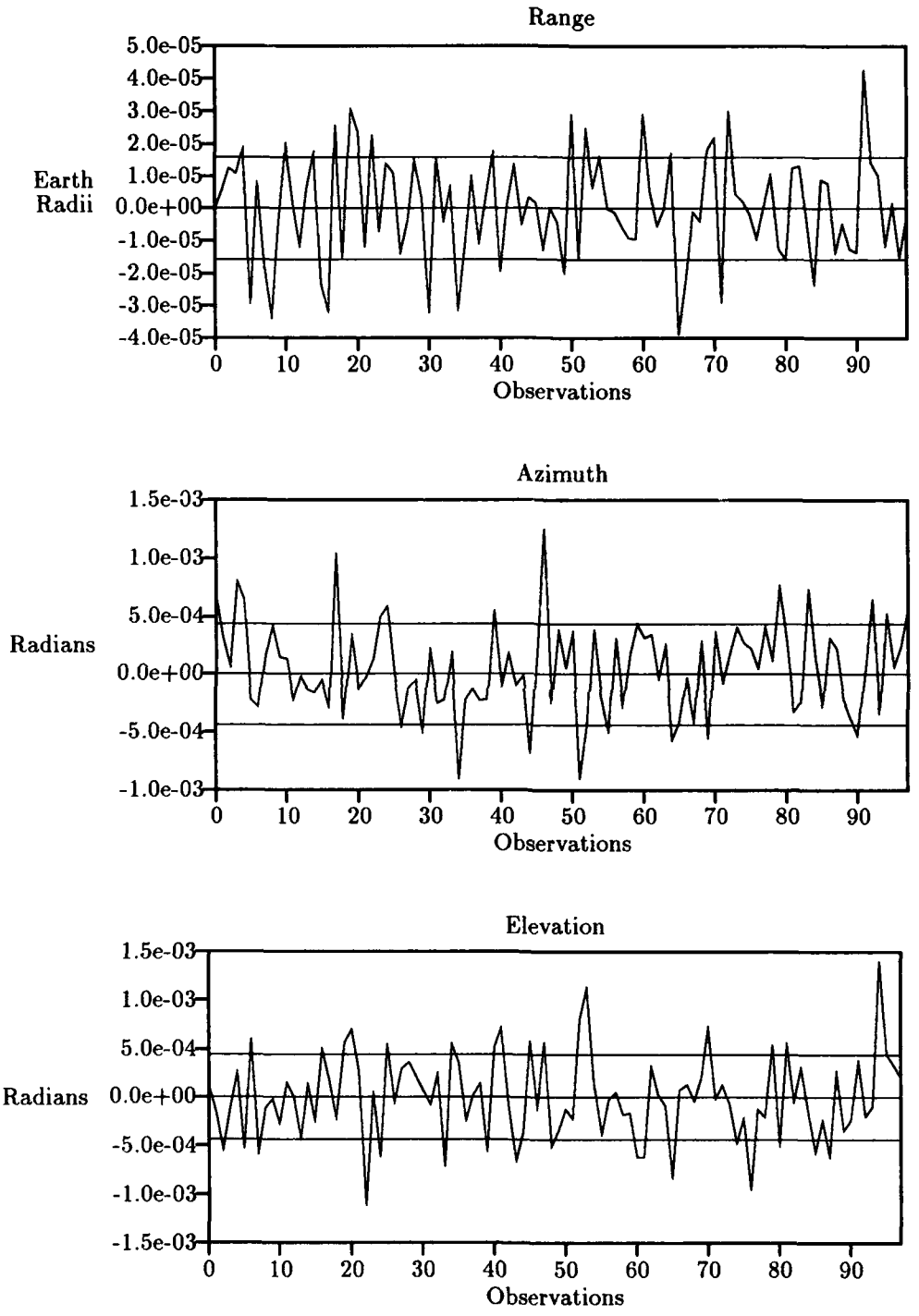


Figure 10. Residual Values versus 1-σ After Convergence — GPS

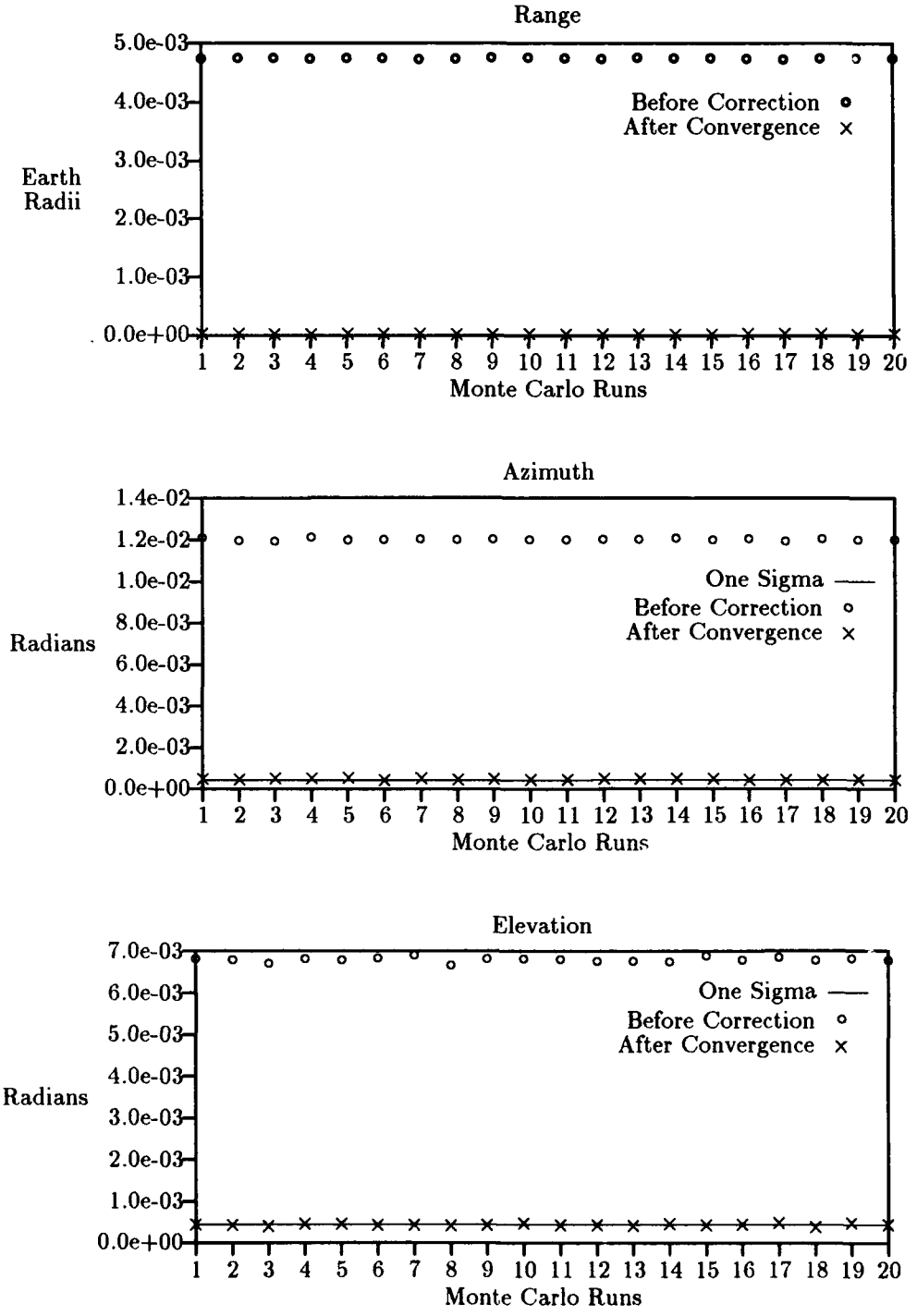


Figure 11. RMS Values Before and After Differential Correction — GPS

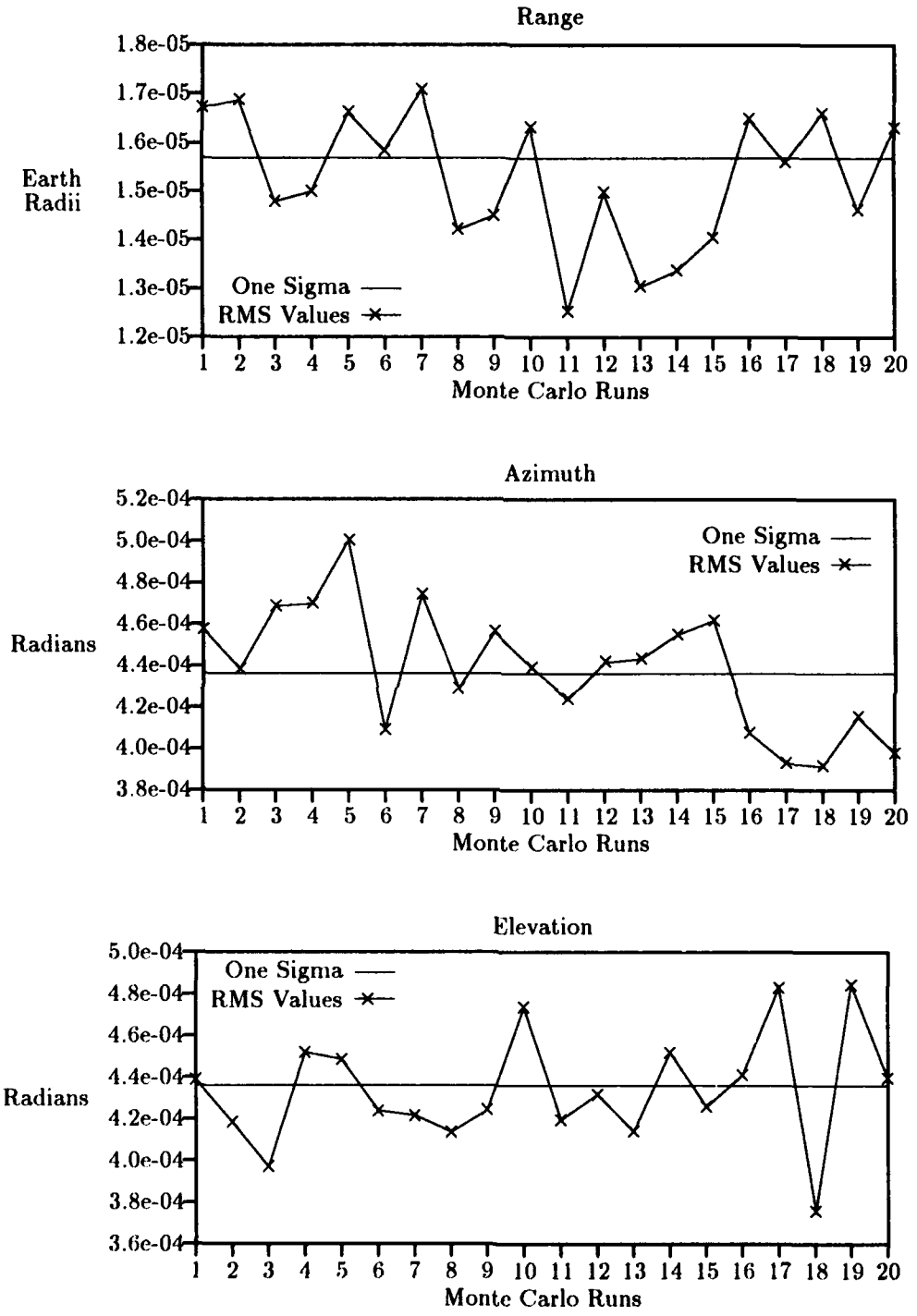


Figure 12. Converged RMS Values versus 1-σ — GPS

4.3.2 Case II — High-Eccentricity/Critical-Inclination Orbit. The second orbit analyzed has a medium altitude and the unique orbital characteristics of high eccentricity and a critical inclination. The critical inclination is the orbit-specific value resulting in no apsidial rotation. The medium-altitude orbits still have relatively long pass lengths. The orbit used is from the the rocket body from Cosmos 1305 launched in 1981 by the USSR (NORAD Catalog Number 12827). The pass considered in this analysis was 2 hours and 47 minutes in duration and used the remote tracking station site REEF (Diego Garcia). There still is a large percentage of the orbit described by the observation data and there also are minimal perturbation effects due to J_2 and air drag. Eccentricity is high, creating a different type of orbit for the equinoctial corrector to estimate. The parameters of the COSMOS orbit are listed in Table 4.

Table 4. Orbit Elements — Cosmos 1305 RB(2)

Cartesian			Classical			Equinoctial		
Epoch	30 Mar 90	09:59:59.67	Per	262.690	min			
x pos	-5444.150	km	a	13586.974	km	ℓ	0.171285	rad
y pos	-5465.509	km	e	0.453789		k	0.398574	
z pos	-0.205652	km	i	63.363	deg	ψ	-0.435544	
x vel	1.769536	km/sec	Ω	225.113	deg	L	1.459538	DU^2/TU
y vel	-3.623977	km/sec	ω	331.441	deg	h	-0.216942	
z vel	7.598636	km/sec	M	9.813919	deg	χ	-0.437265	
			alt	7208.837	km			

The REEF pass examined was on 01 April 1990 from 06:40–09:27, approximately two days beyond the epoch time with the observations spaced at 60-second increments. Larger time increments were spaced too far apart for the Hamming routine in the truth model to initialize. This is most likely due to the high eccentricity of the orbit. The two-day span of time from epoch is still a reasonable time to expect a remote site to use a reference. As in Case I, Case II uses only a two-body only integrator to propagate the reference trajectory from the epoch time to the time of the first observation.

This estimation attempt used the same σ multiplier as in Case I of 3.0×10^5 and achieved adequate results. Figure 13 shows the overall corrector performance for the 168 data points in this pass. The overall view once again shows on a large scale that the residuals were reduced. The close-up view in Figure 14 shows the final residual values after convergence in relation to the respective σ values. The overall trend in the data shows that the residuals are near the one sigma success criteria.

Figure 15 shows the overall performance of reducing the initial RMS values to near the desired one-sigma level (the previous figures show the results for Run 9). Figure 16 then shows the detailed view of the final RMS values. Note again that there are final RMS values that are both above and below the one-sigma line but the overall average tends to follow the desired limit.

The twenty Cosmos runs show that the equinoctial differential corrector is effective for high-eccentricity/critical-inclination orbits with perturbation effects. However, the data must be closely monitored to ensure that the rejection criteria is set at an appropriate level to edit the large residuals while passing the small residuals. The residual computation had to be modified to handle the azimuth change from near 0 degrees to values near 360 degrees. Once this change was in effect, the large azimuth residuals disappeared and normal results were output. As in Case I, this orbit can be corrected with fewer than 10 data points and still converge.

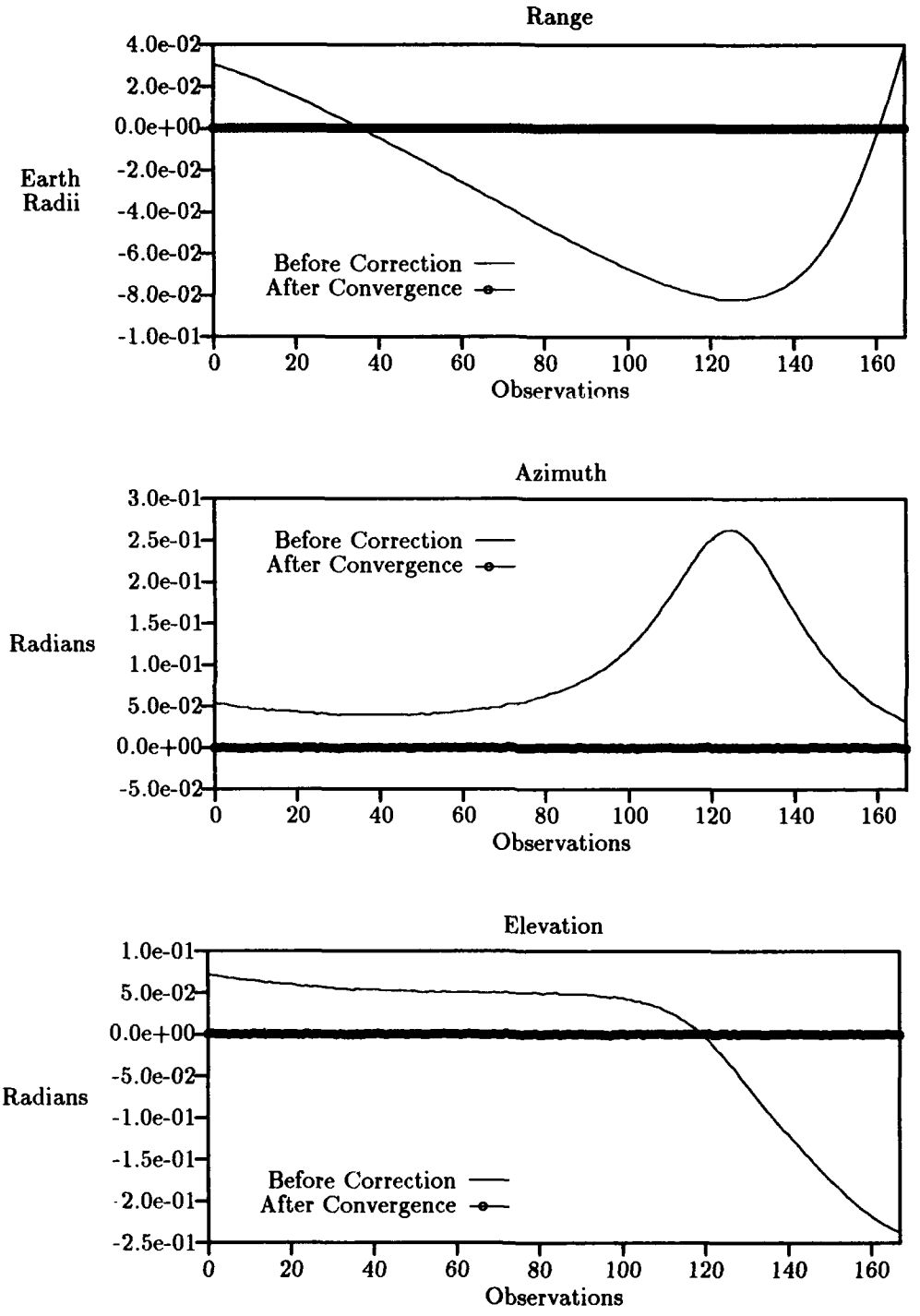


Figure 13. Residual Values Before and After Differential Correction — Cosmos

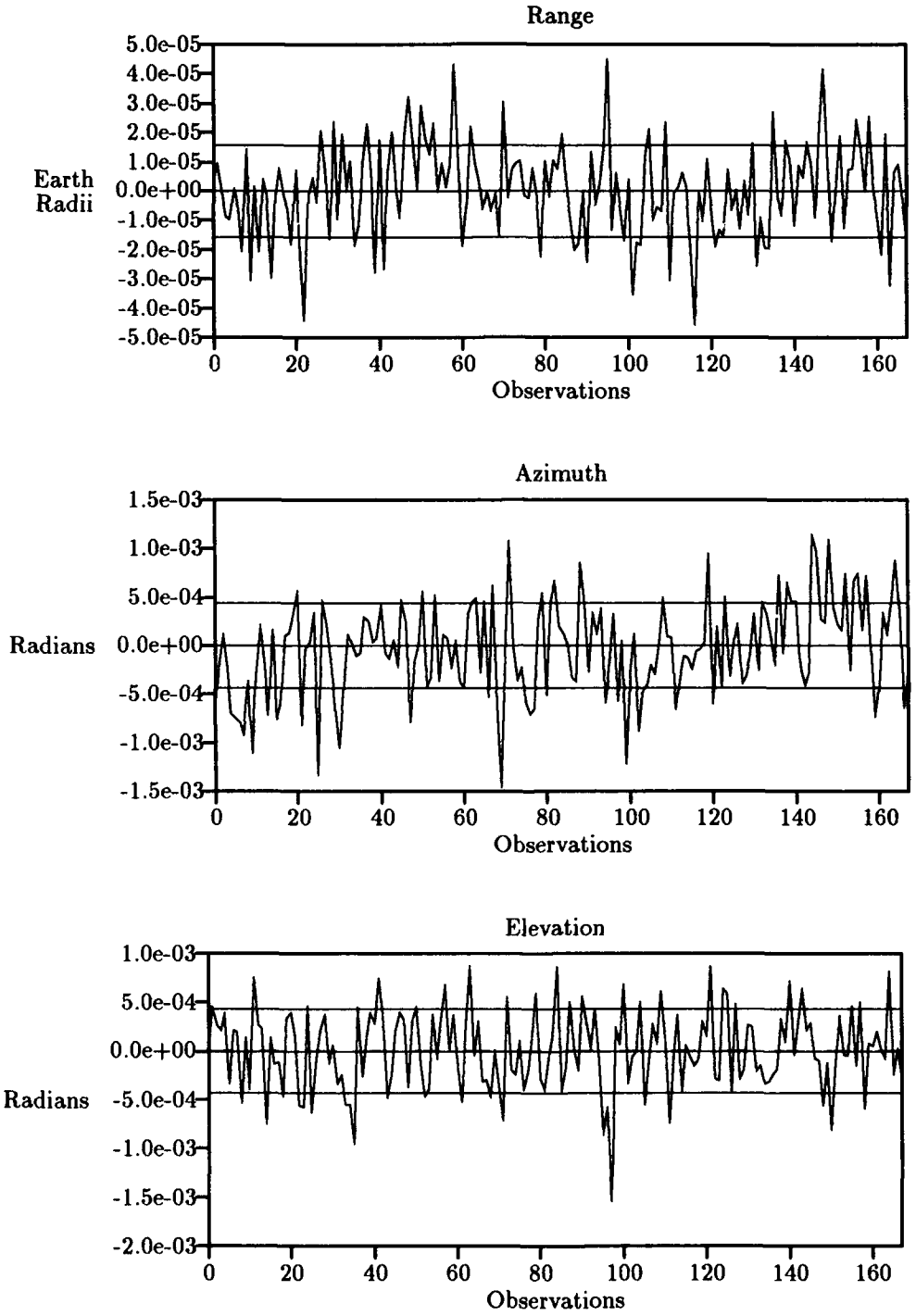


Figure 14. Residual Values versus 1-σ After Convergence — Cosmos

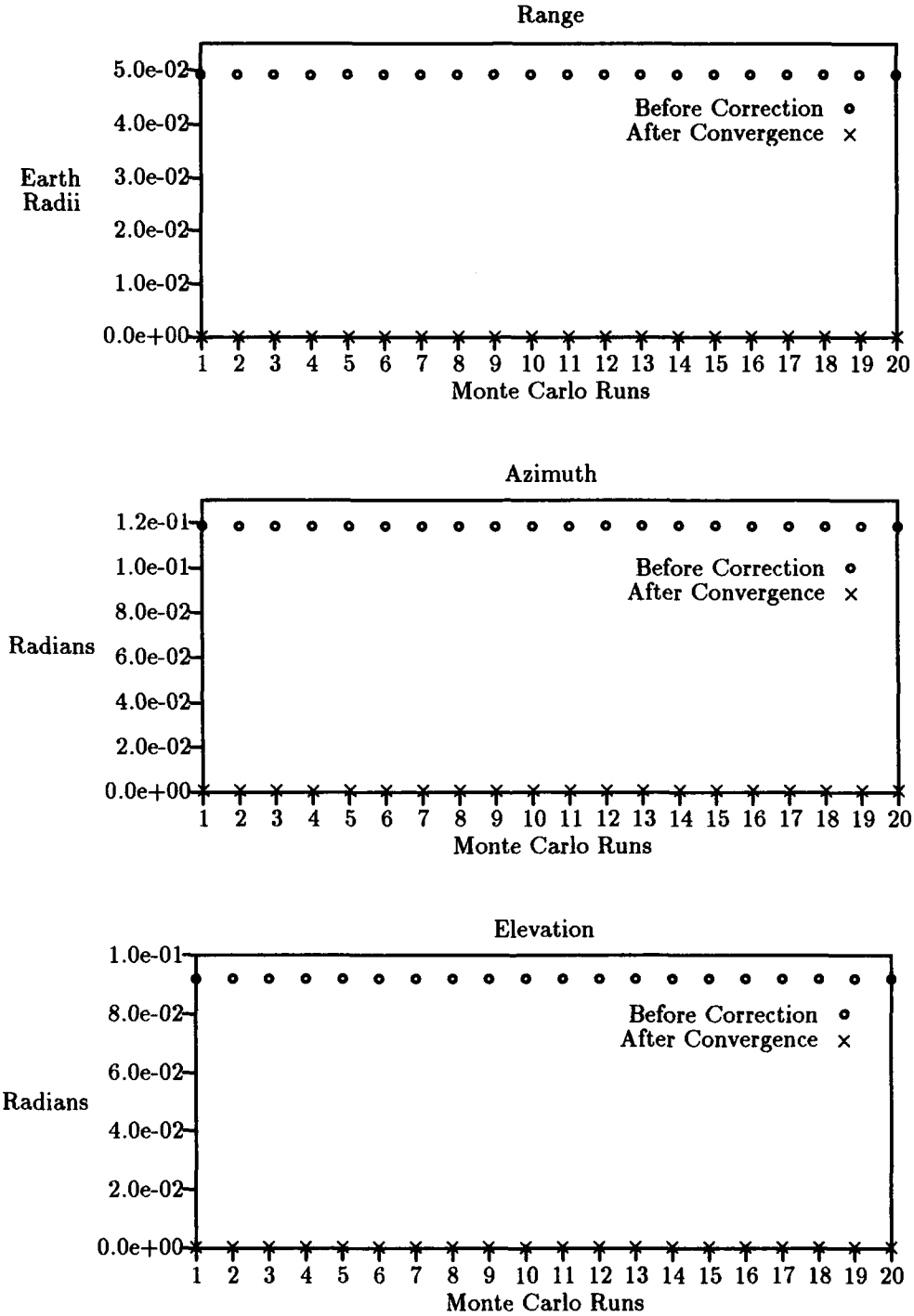


Figure 15. RMS Values Before and After Differential Correction — Cosmos

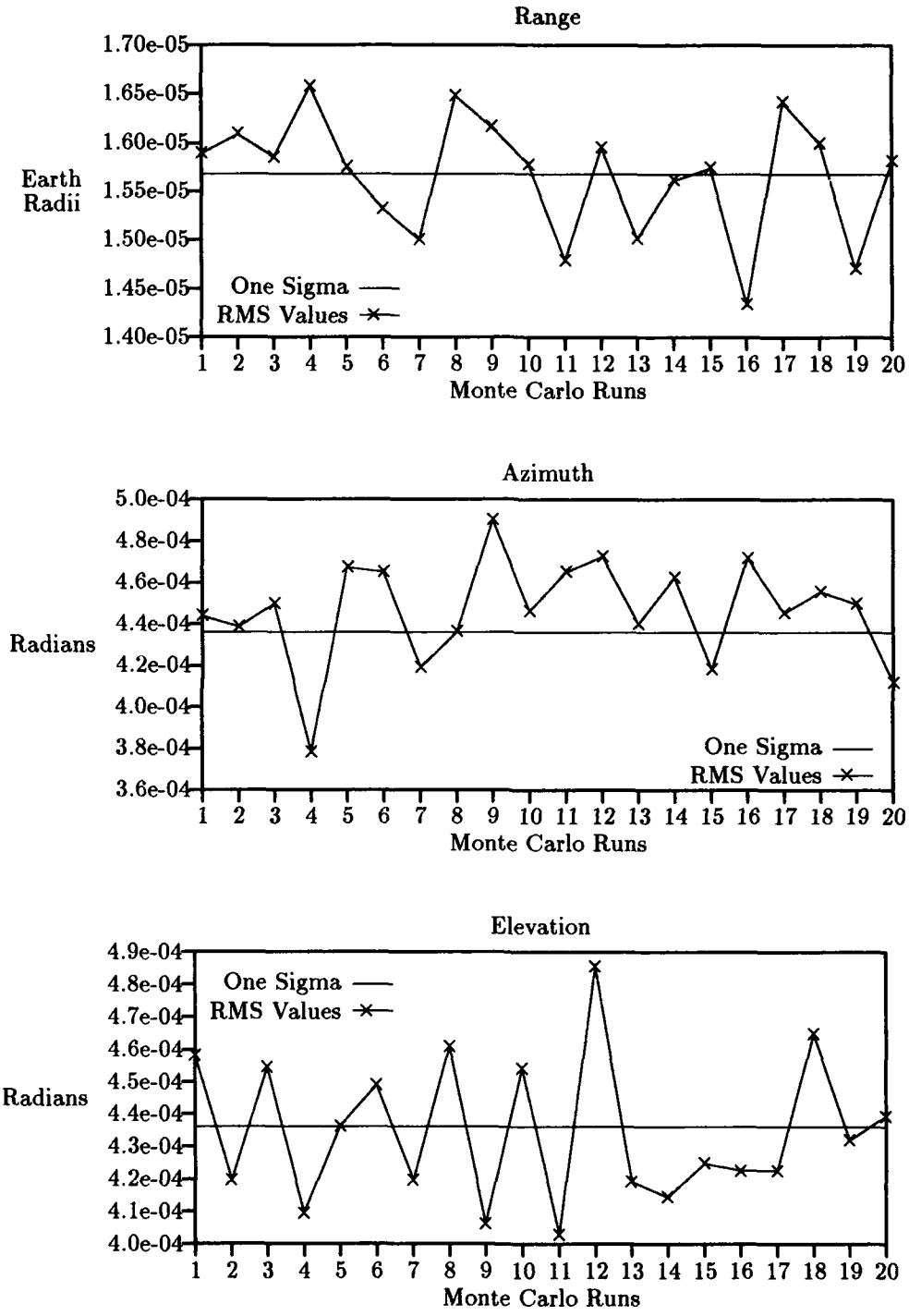


Figure 16. Converged RMS Values versus 1-σ — Cosmos

4.3.3 Case III — Medium-Altitude/High-Inclination Orbit. The third orbit analyzed has a medium altitude and a very high inclination of 120 degrees. This altitude orbit also has relatively long pass durations. The orbit used is from the debris of the Explorer satellite launched in 1968 by the US (NORAD Catalog number 04841). The passes considered in this analysis were 39, 46, and 42 minutes in duration and used the remote tracking station site GUAM. The J_2 perturbation begins to have a significant effect on the orbit in this case. The nodal rotation at this altitude and inclination is approximately 1.3 degrees per day. Air drag is not a factor yet with the altitude greater than 1000 kilometers. The parameters of the Explorer orbit are listed in Table 5.

Table 5. Orbit Elements — Explorer Debris

Cartesian			Classical			Equinoctial		
Epoch	15 Mar 90	02:37:30.63	Per	155.516	min			
x pos	8259.152	km	a	9579.522	km	ℓ	1.024542	rad
y pos	-2896.093	km	e	0.271009		k	0.049183	
z pos	1287.749	km	i	120.737	deg	ψ	1.703580	
x vel	-0.244773	km/sec	Ω	345.696	deg	L	1.225535	DU^2/TU
y vel	-3.595045	km/sec	ω	280.456	deg	h	-0.266508	
z vel	5.960016	km/sec	M	58.70197	deg	χ	-0.434364	
			alt	3201.385	km			

The first GUAM pass analyzed spanned 40 points on 17 March 1990 from 01:05–01:44, which is two days past epoch. The time increment of the data points is 60 seconds. A unique feature of the observation data in this pass is a very high maximum elevation, near 85 degrees, which causes rapid changes in azimuth as the satellite passes overhead. Figure 17 shows large discontinuities in both the azimuth and elevation residuals. With these fluctuations in the residual values, the differential correction process converged in six iterations with the usual initial σ multiplier of 3.0×10^5. Attempts to remove the bad data by lowering the multiplier value to 3.0×10^3 resulted in singular matrix inversion problems.

The large fluctuations in the residuals in Pass 1 can be eliminated. These errors are reduced by including the full integrator with perturbations in the least squares routine. This generates a

more accurate description of the orbit described by the data. Figure 18 shows the new before and after correction residual values with the updated integrator. The estimator now easily converges to the final values in three iterations, seen in detail in Figure 19. Using the full integrator for this case is justified in order to eliminate the chance of a poor reference trajectory giving high residual values.

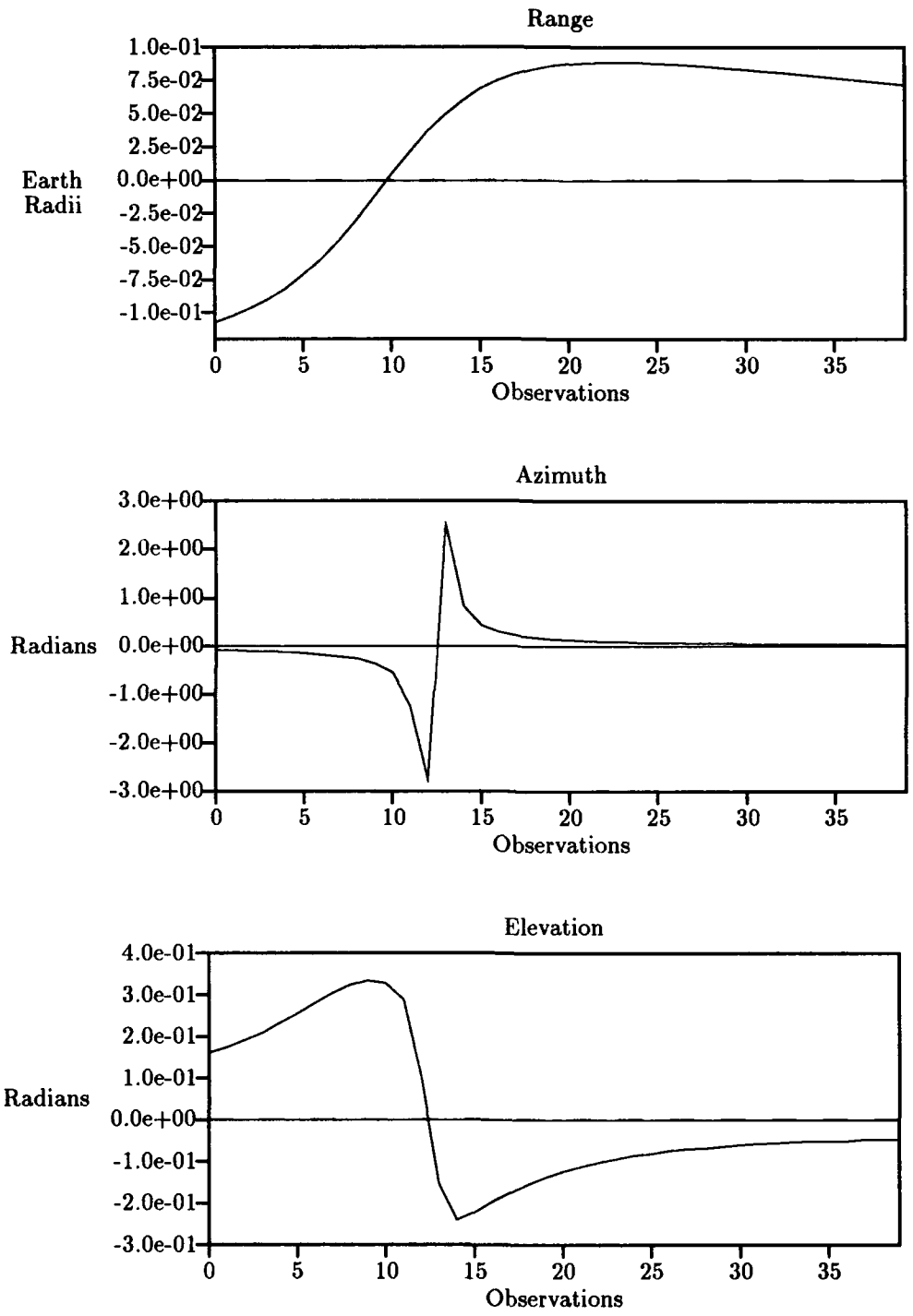


Figure 17. Residuals Before Correction, 2-Body Integrator — Explorer (Pass 1)

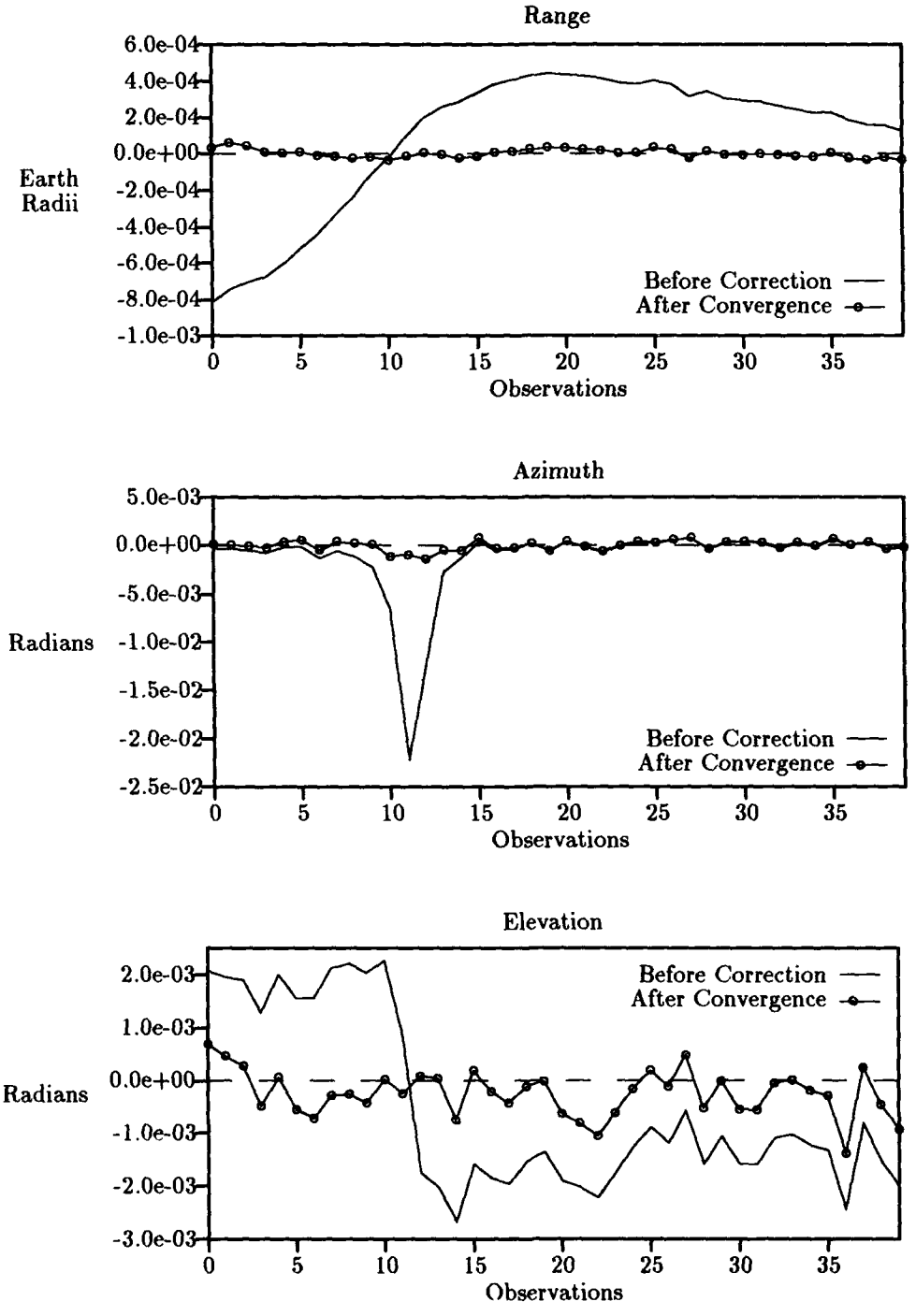


Figure 18. Residual Values with Full Integrator (Pass 1)

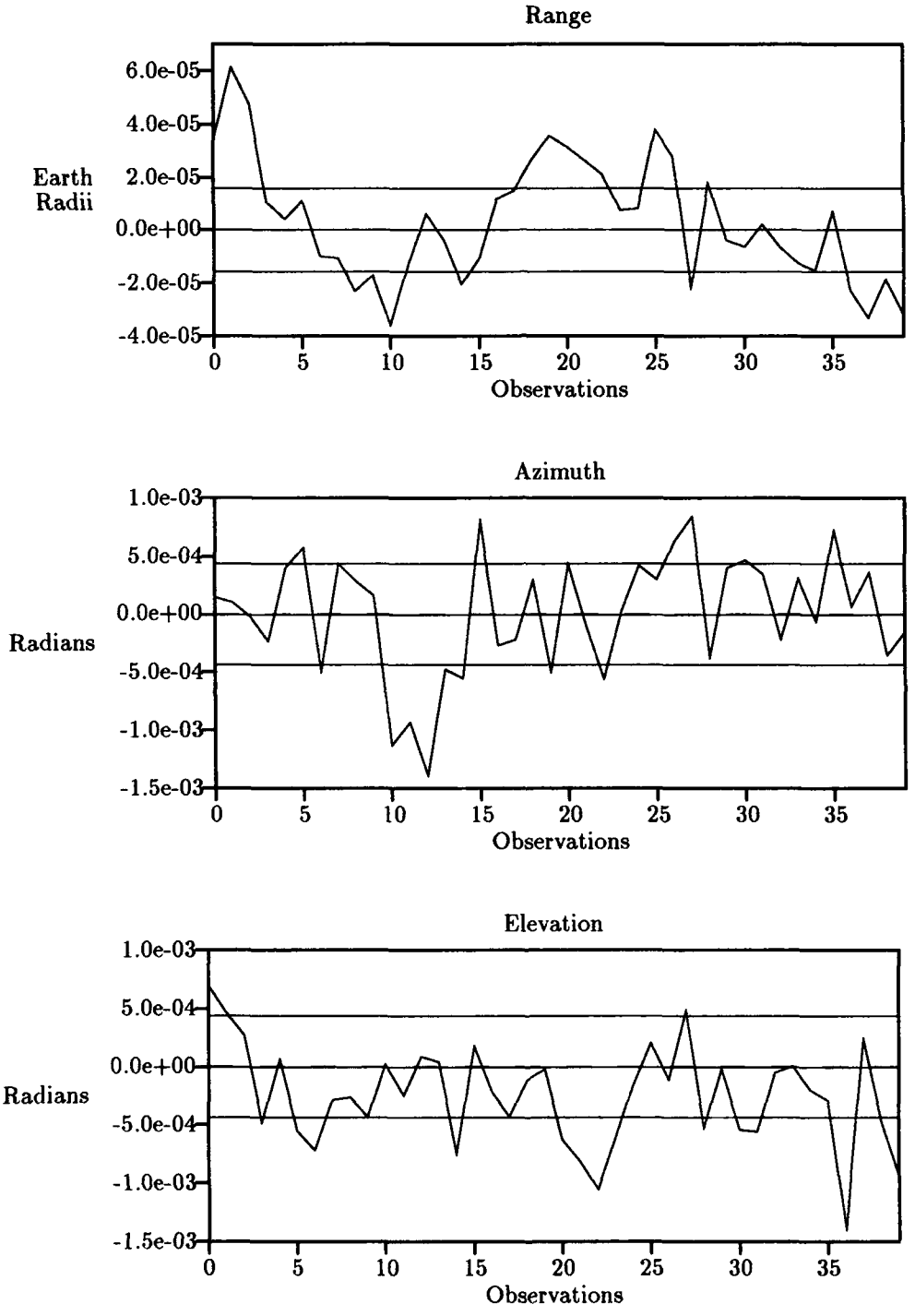


Figure 19. Residuals versus 1-σ After Convergence — Explorer (Pass 1)

The next GUAM pass looked at was one day past epoch on 16 March 1990 from 13:20–14:06 again at 60-second increments. The characteristics of this pass are tame compared to the previous pass in this case. The maximum elevation is near 69 degrees and there are no rapid changes in either the elevation or the azimuth observations. One day from epoch is a very realistic time frame in which tracking stations would use a reference trajectory. The orbit is still sufficiently deterministic to use only two-body propagation of the reference state to the time of the first observation. The purpose of switching to a new pass is to prove the estimators capabilities at a lower altitude than in previous cases without throwing the estimator off by handling a separate high elevation problem.

The estimator worked fine with an initial rejection criteria of 3.0×10^5. From Figure 20, notice the typical patterns of large azimuth residual values. The peak error occurs once again at the highest elevation values which cause the most rapid changes in azimuth. The elevation residual curve also shows this point where the residual values change sign. This represents the transition from rising elevation observations to falling values. The overall view of the 47 points from this pass, shown in Figure 21, indicates that the residuals have been sufficiently reduced below the one-sigma error value. Yet, as shown in subsequent figures, the overall performance for this pass is not as good as in previous cases.

While Figure 22 shows the overall performance of reducing the initial RMS values to below one sigma, Figure 23 shows the detailed view of the final RMS values which have a trend that is greater than the sigma error level (The previous figures show the results for Run 13). It appears that only the azimuth correction was performed to the desired standard. This may be due to the lower altitude, shorter pass length data set, or possibly due to other factors that will be examined.

Now, as in Pass 1, it might be possible to improve the overall RMS values by including all perturbation effects in the initial propagation of the reference state. Because the Explorer orbit is affected by the J_2 harmonic, including these effects in the integrator will make the reference propagation a better approximation to the actual data and hopefully reduce the associated errors.

From Figure 24, the initial values of the residuals are reduced by significant amounts from the two-body integrator case and the number of iterations is reduced from five to three. Nevertheless, the post correction values with full integrator are identical as before. Therefore, it provides no advantage to change the integrator in Pass 2 when the correction is capable of being performed in the first place.

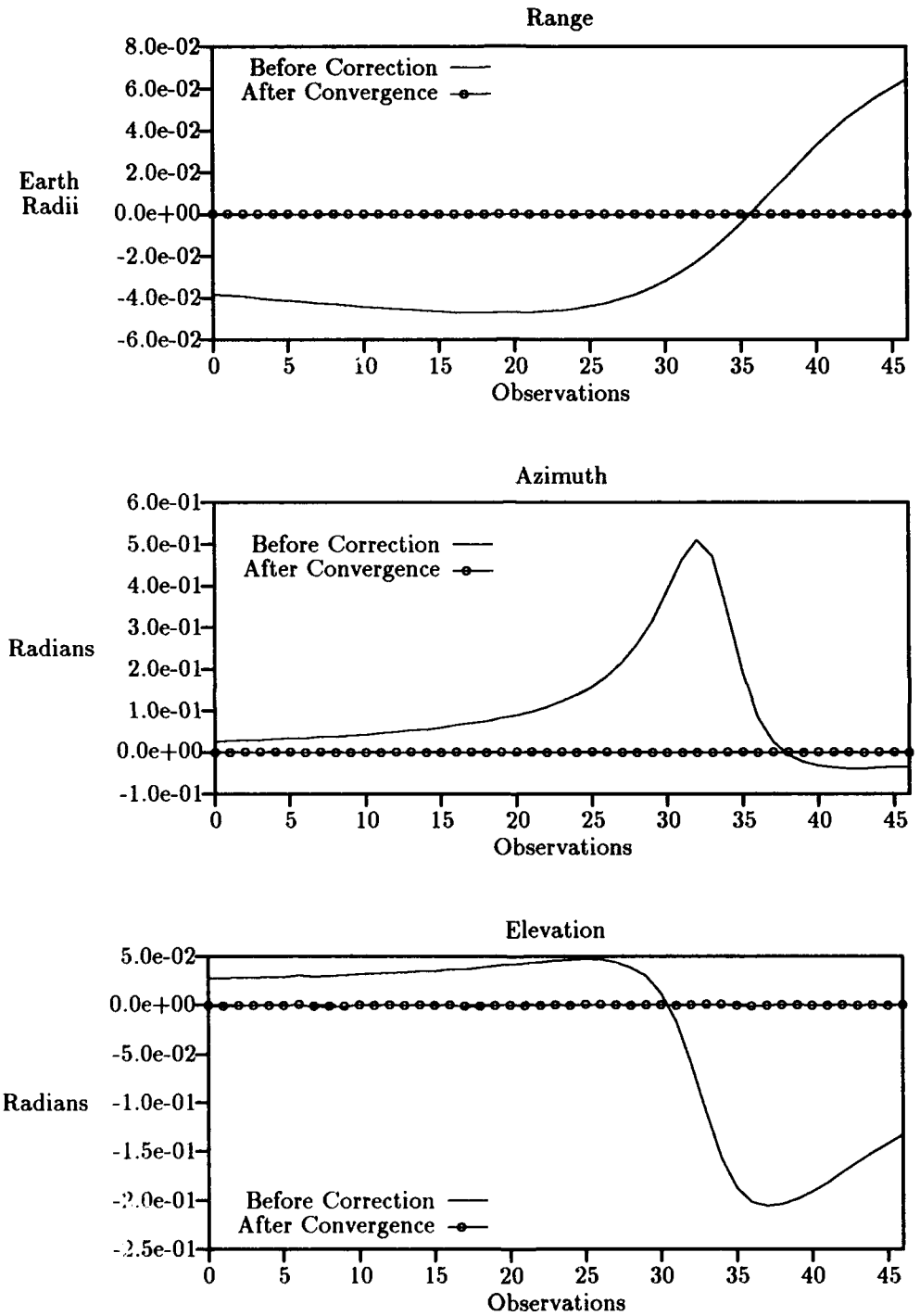


Figure 20. Residuals Before and After DC — Explorer (Pass 2)

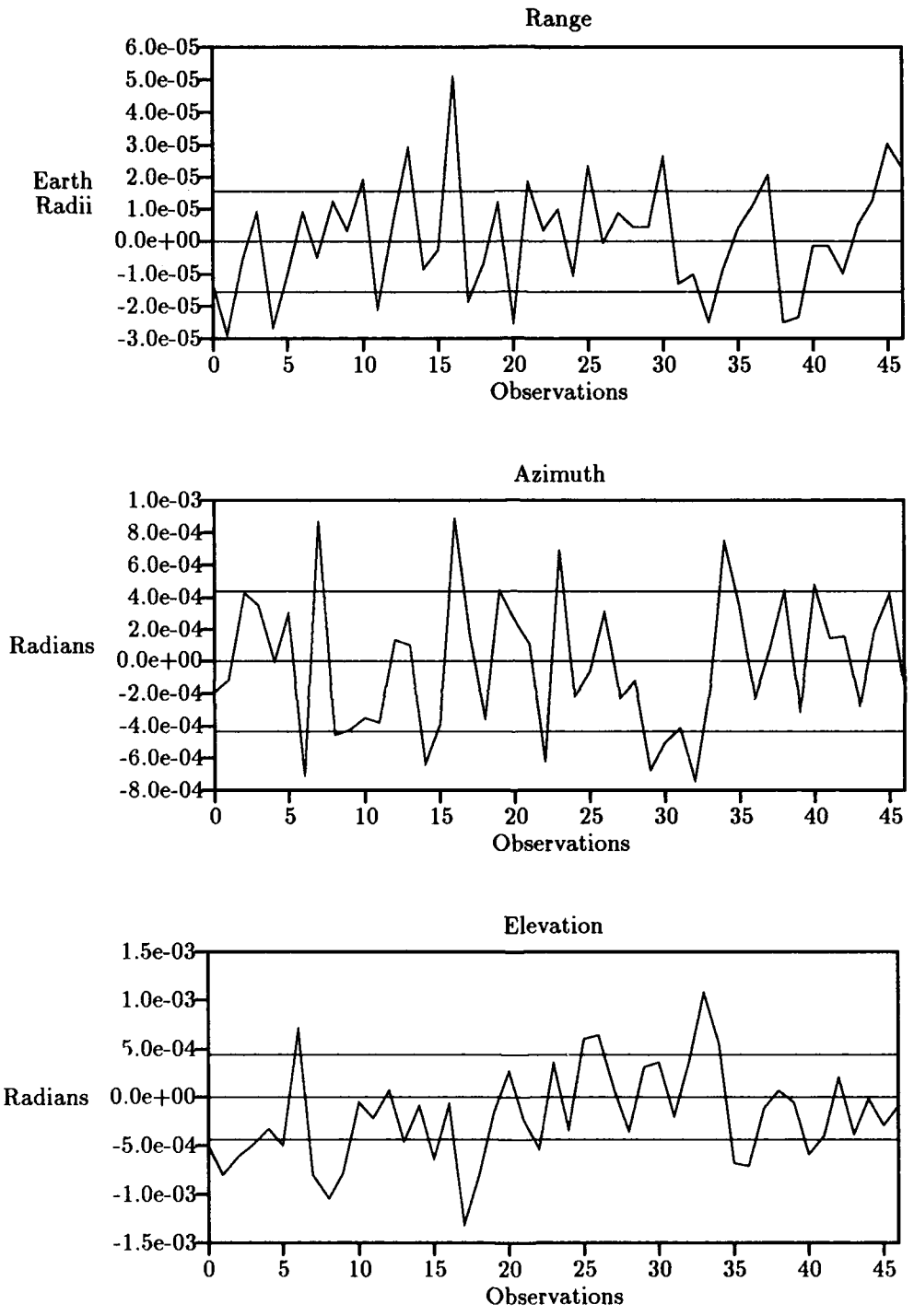
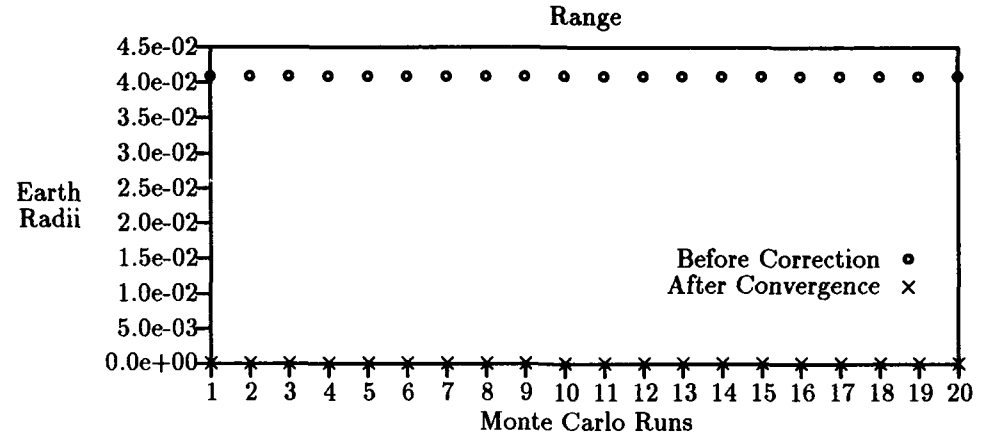
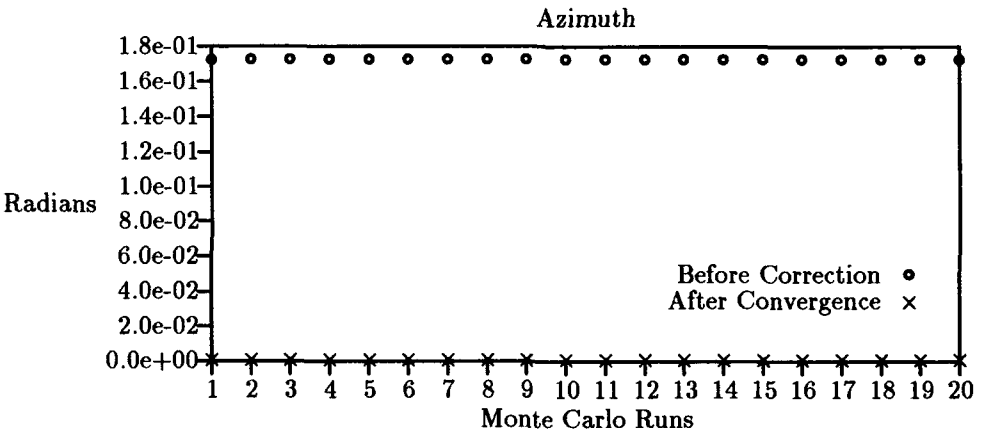


Figure 21. Residuals versus 1-σ After Convergence — Explorer (Pass 2)





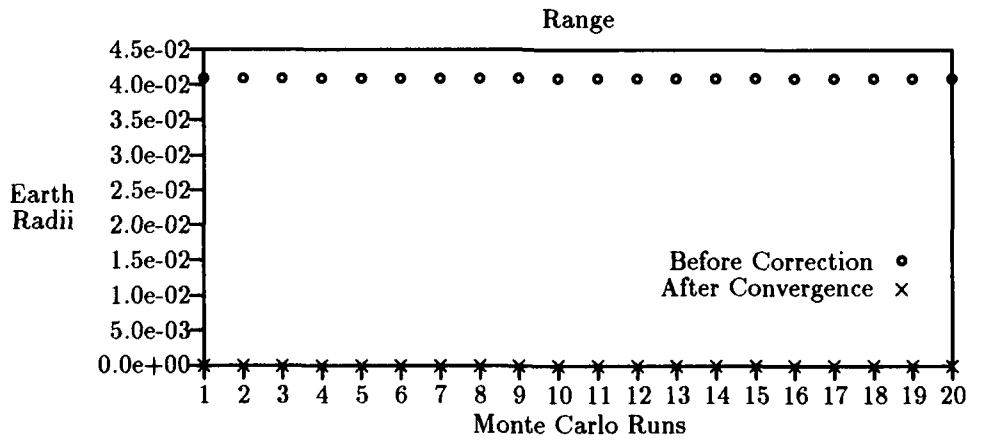


Figure 22. RMS Values Before and After Differential Correction — Explorer

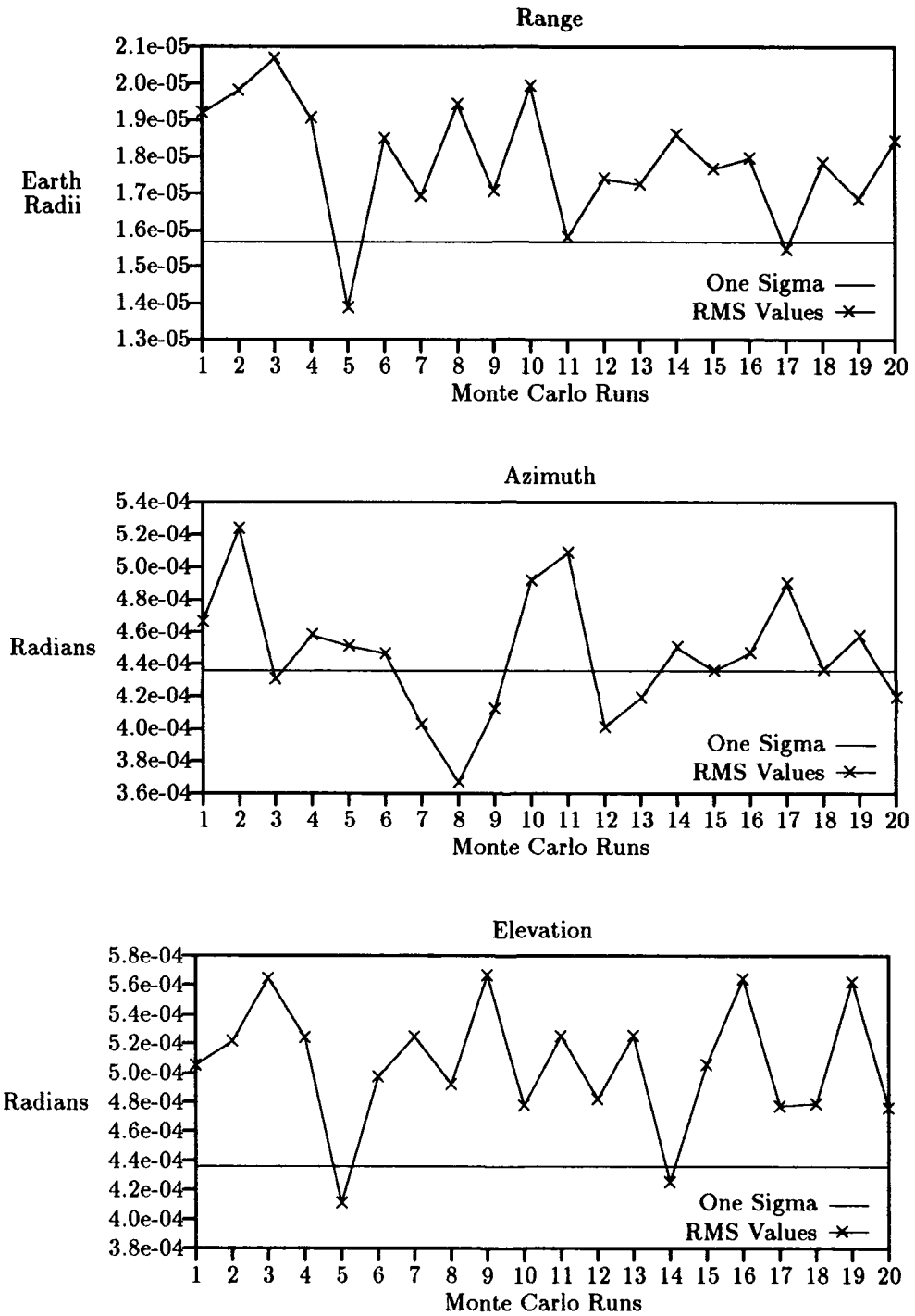


Figure 23. Converged RMS Values versus 1-σ — Explorer

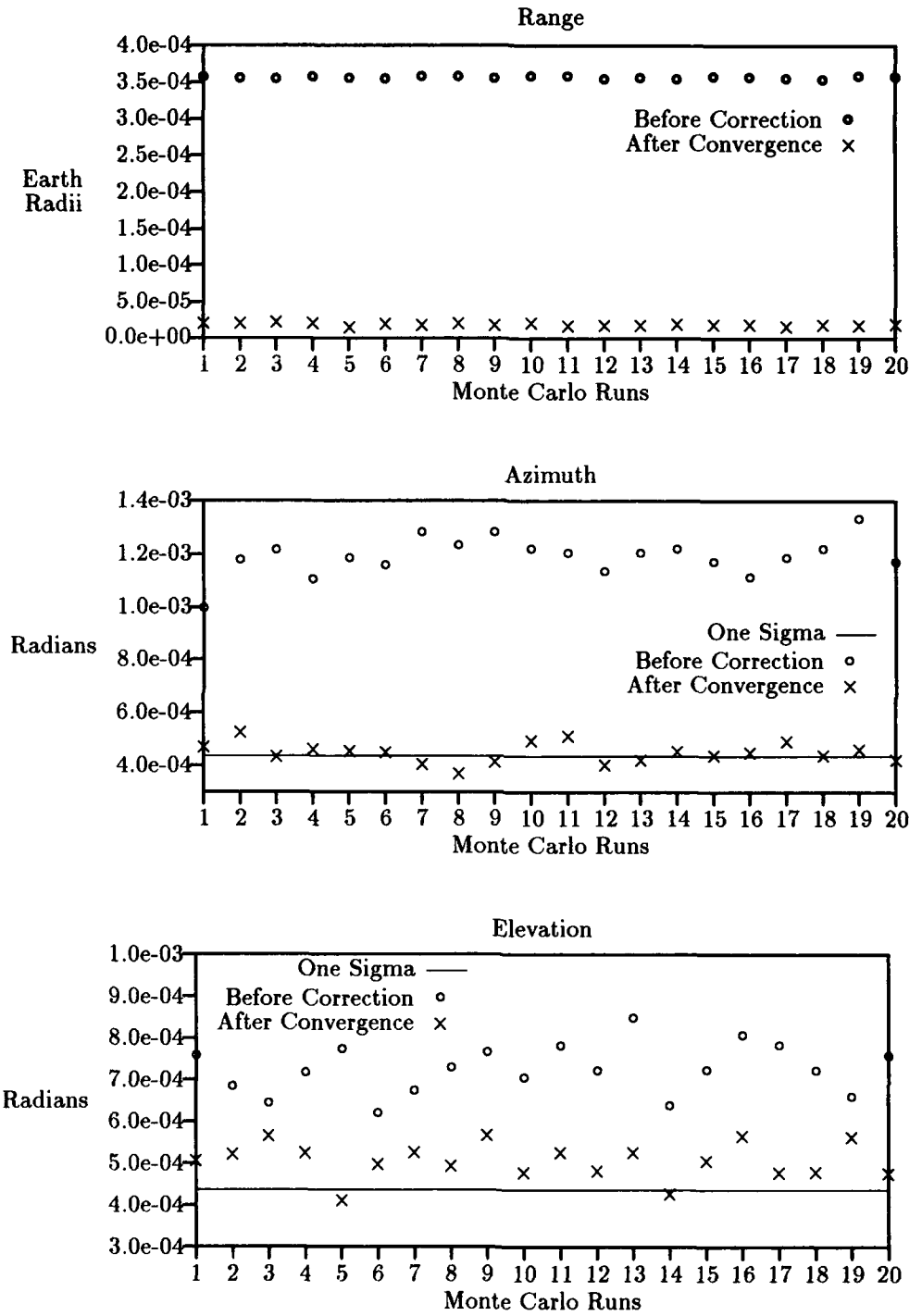


Figure 24. RMS Values Before and After Differential Correction — Full Integrator

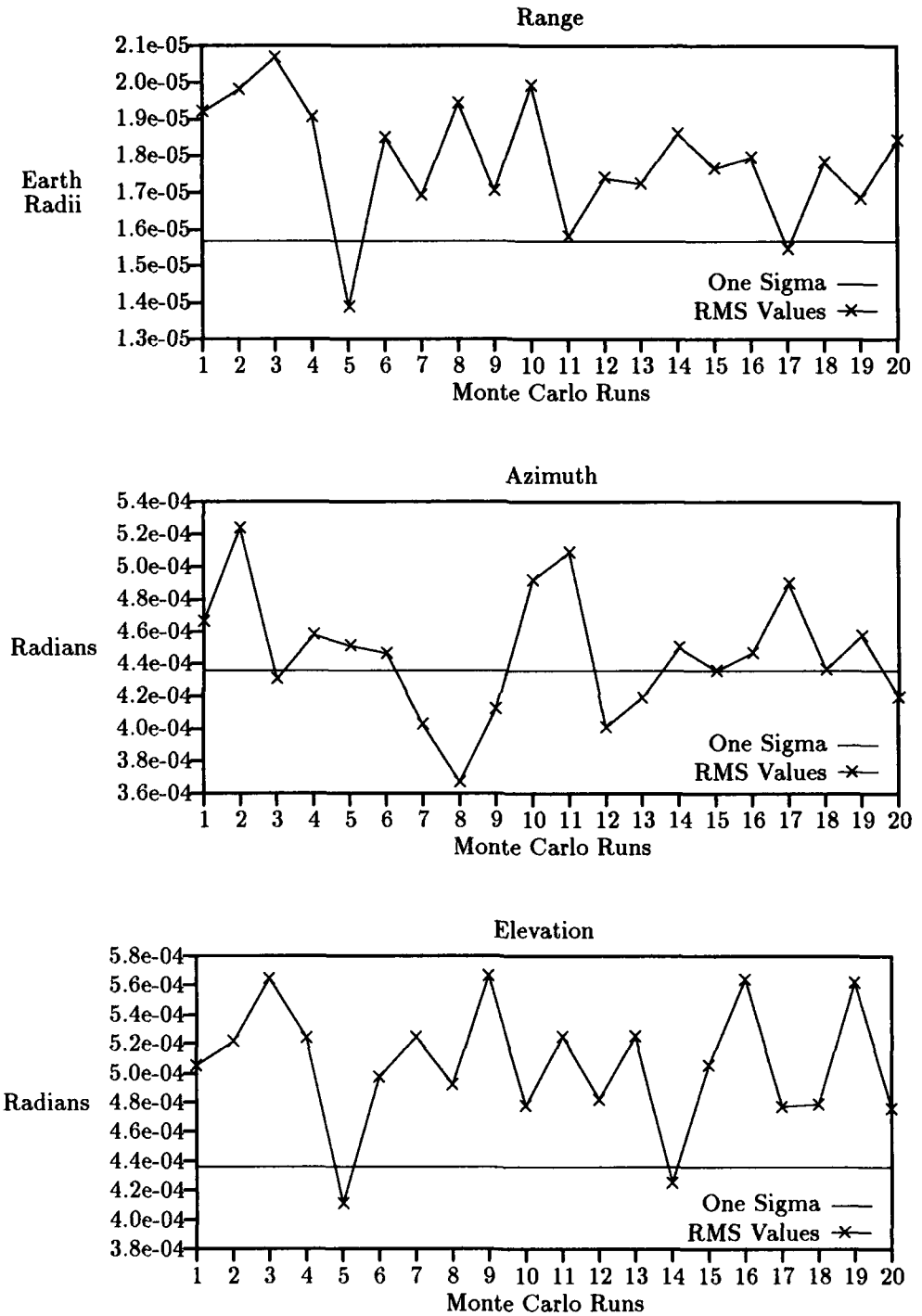


Figure 25. Converged RMS Values versus 1-σ — Full Integrator

The next attempt at improving the overall converged RMS values is to tighten the rejection criteria. Earlier attempts to lower the multiplier have been ineffective and this case is no different. By lowering the rejection criteria, there wasn't any help at all in rejecting larger residuals for a multiplier of 3.0×10^4 and smaller multipliers resulted in singular matrix inversions.

Most of the errors involved in this case center near high elevation data points within the pass. Figure 26 shows the results from choosing a pass with a low maximum elevation of 28.7 degrees. The pass is on 16 March 90 from 22:39–23:21 and has 43 points. As indicated in the figure, lower elevations with a slower change in both azimuth and elevation observations allow the estimator to make a better attempt at predicting the reference trajectory resulting in smaller residuals and overall RMS values.

The last attempt to improve the converged RMS values combined two of the previous passes. This increases the total number of points used in the correction and allows for slightly different looks at the same arc of data. Unfortunately, the RMS values after convergence were a factor of 10 higher than the single pass case. The corrector was able to process the data and converge on a solution, but it was no where near the one sigma error level desired.

The different Explorer runs show that the equinoctial differential corrector is effective for medium-altitude/high-inclination orbits with perturbation effects only when the maximum elevation is low. Attempts to improve the converged RMS values by changing the initial integrator to include perturbations improved the initial RMS values but did not help the converged values. Multiple passes and tighter σ multiplier rejection criteria did not improve the estimator's performance.

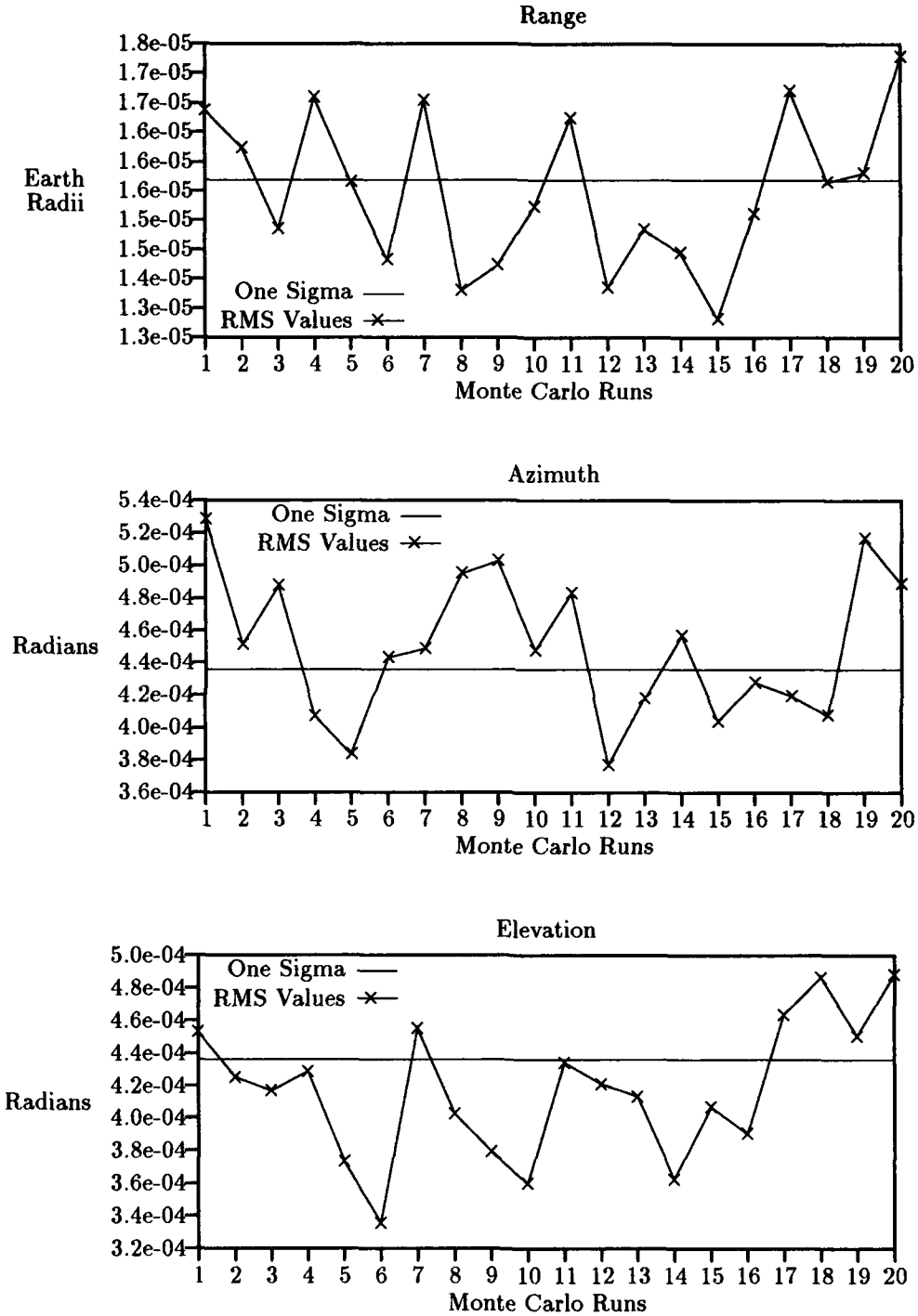


Figure 26. Converged RMS Values versus 1-σ — Low Max Elevation

4.3.4 Case IV — Low-Altitude/Sun-Synchronous Orbit. The fourth orbit analyzed has a low altitude and is the sun-synchronous weather satellite, DMSP. This altitude orbit also has short pass lengths. The pass considered in this case is 13.5 minutes in duration and uses the remote tracking station site POGO (Thule). The J_2 perturbation has a significant effect on the orbit and air drag also becomes a factor. The nodal rotation for a sun-synchronous orbit is approximately 0.985 degrees per day. The parameters of the DMSP orbit are listed in Table 6.

Table 6. Orbit Elements — DMSP

Cartesian			Classical			Equinoctial		
Epoch	10 Sep 92	10:12:00	Per	101.808	min			
x pos	-156.876	km	a	7222.392	km	ℓ	0.042902	rad
y pos	-6476.819	km	e	0.001076		k	-0.000942	
z pos	3174.432	km	i	98.797	deg	ψ	0.116607	
x vel	-1.344282	km/sec	Ω	84.264	deg	L	1.064129	DU^2/TU
y vel	-3.193152	km/sec	ω	151.098	deg	h	0.000520	
z vel	-6.580665	km/sec	M	2.458131	deg	χ	1.160818	
			alt	844.255	km			

The POGO pass tested has 27 points spaced 30 seconds apart on 10 September 1992 from 13:14:30–13:27:00. The corrector performance for this low altitude orbit was inadequate. Several attempts to converge to a solution all resulted in failure. The first attempt used a pass very close to the original epoch time to reduce the initial propagation. The corrector diverged by systematically making corrections that were too large, maximizing the error rather than minimizing the error. Although Figure 27 does not indicate any problem areas with the initial residuals, both the two-body only and the full integrator failed.

The next attempt to coerce the corrector to converge was to simplify the problem by correcting only the in-track elements of the mean anomaly term, ℓ, and the mean motion term, L. Theoretically, this would remove all of the initial in-track errors allowing the corrector to deal with only the other, harder to determine, four terms remaining on a second loop through the corrector. The process did make the initial correction to the two in-track terms, but the changes were small

enough to be insignificant in helping the main iteration correction converge. It made no difference to correct any two elements first and then enter into the full correction.

Examining the source of divergence can be accomplished by using the advantages of the truth model. By removing the perturbation effects, the "perfect" data can indicate where the problem lies. Removing both the J_2 and air drag perturbations made a dramatic effect — the corrector converged in six iterations. Processing data with the J_2 perturbation alone caused divergence as in the original case. Processing data with the air drag perturbation alone converged to a solution. At an altitude very high in the air drag regime, this should have a minimal effect. Therefore, it seems that the source of the problem at the lower altitude lies with the J_2 perturbation alone.

Another area of concern for the low altitude orbit is the lack of data points generated by such a quick pass. Examining this area with the convergent cases of the corrector (with or without air drag), there was no indication that the corrector could not handle just a few data points. This is consistent with the other orbit cases in the study which had convergent solutions with as little as two data points. The divergence is associated with the perturbation effects and not due to the lack of data.

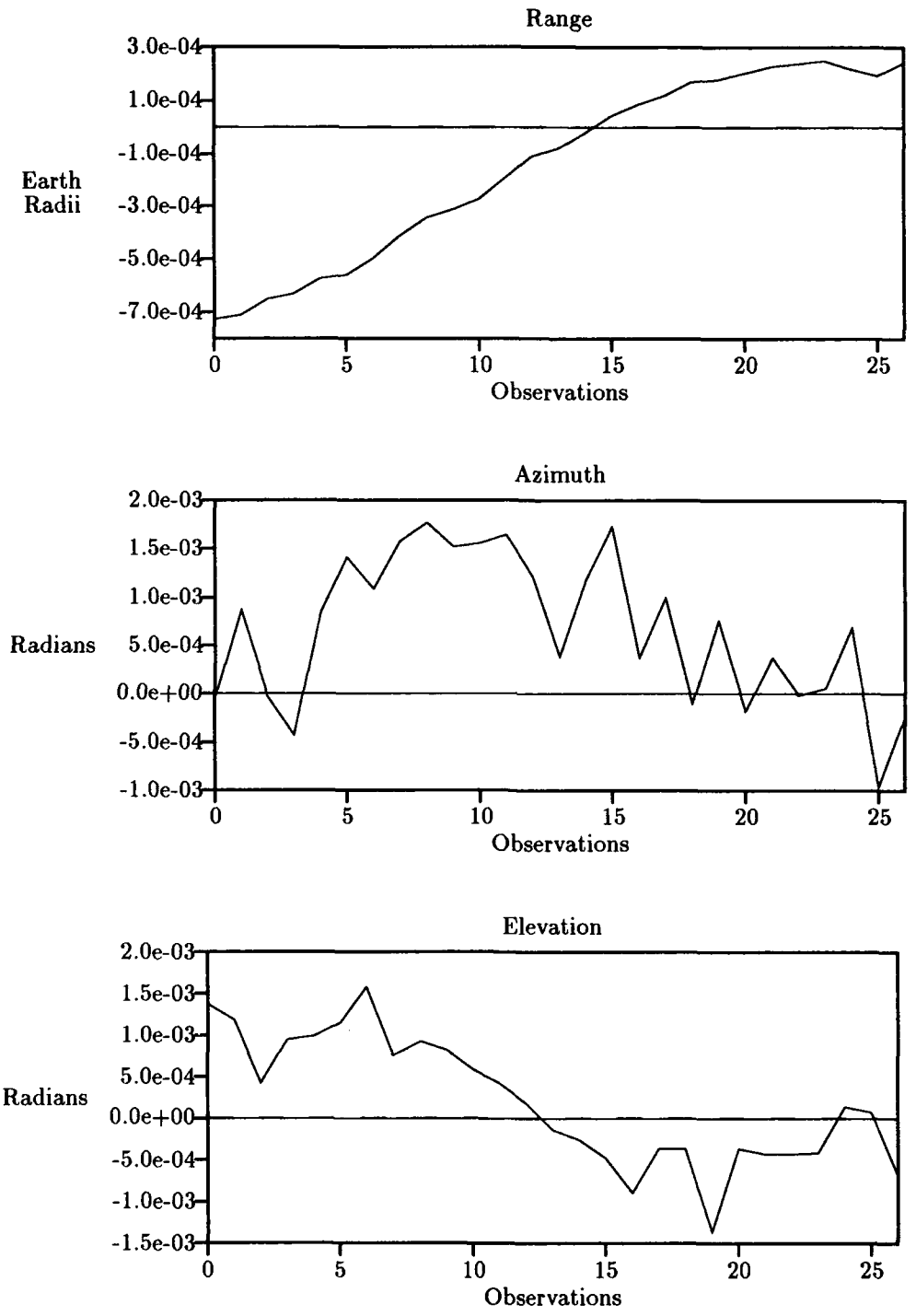


Figure 27. Residual Values Before Differential Correction — DMSP

4.3.5 Case V — Low-Altitude/Low-Eccentricity Orbit. The last orbit analyzed has the low altitude and low eccentricity of the Russian Mir space station. This altitude orbit has very short pass lengths of approximately ten minutes. The pass considered in this case is 9.5 minutes in duration and uses the remote tracking station site GUAM. The J_2 perturbation has a significant effect on the orbit and air drag is a significant factor. The parameters of the Mir orbit are listed in Table 7.

Table 7. Orbit Elements — Mir

Cartesian			Classical			Equinoctial		
Epoch	10 Sep 92	10:12:00	Per	92.699	min			
x pos	5097.638	km	a	6784.906	km	ℓ	0.660860	rad
y pos	-2716.526	km	e	0.001504		k	-0.000266	
z pos	3544.054	km	i	51.625	deg	ψ	-0.483613	
x vel	5.060657	km/sec	Ω	181.016	deg	L	1.031397	DU^2/TU
y vel	3.636431	km/sec	ω	100.188	deg	h	0.001481	
z vel	-4.478165	km/sec	M	37.86446	deg	χ	-0.008574	
			alt	406.769	km			

The GUAM pass tested has 39 points spaced 15 seconds apart on 10 September 1992 from 13:32:30–13:42:00. The corrector performance for this low-altitude orbit was also inadequate. As in Case IV, several attempts to converge to a solution all resulted in failure. The first attempt used a pass very close to the original epoch time to reduce the initial propagation. The corrector diverged by making the wrong corrections, maximizing the error. The farther away from the initial epoch time, the faster the corrector diverged. Once again, Figure 28 does not indicate any problem areas with the initial residuals. Neither the two-body only nor the full integrator was able to converge upon a solution with this orbit.

Correcting only the in-track elements of the mean anomaly term, ℓ, and the mean motion term, L, did not improve performance. As in Case IV, it took the same number of iterations to diverge in the full correction run as in the original attempt. The process did make the initial correction to the two in-track terms, but again the changes were too small to be significant in

helping the main iteration correction converge. It made no difference to correct any two elements first and then enter into the full correction for this orbit case.

Removing the J_2 term alone allowed the corrector to converge in six iterations. The air drag term was not large enough to keep the corrector from performing as expected. At such a low altitude, the air drag term is near a maximum density value. However, even at this maximum value, the air drag term is six orders of magnitude smaller than the J_2 term. Therefore, it is obvious that the harmonic term has the dominant perturbation effect on the orbit.

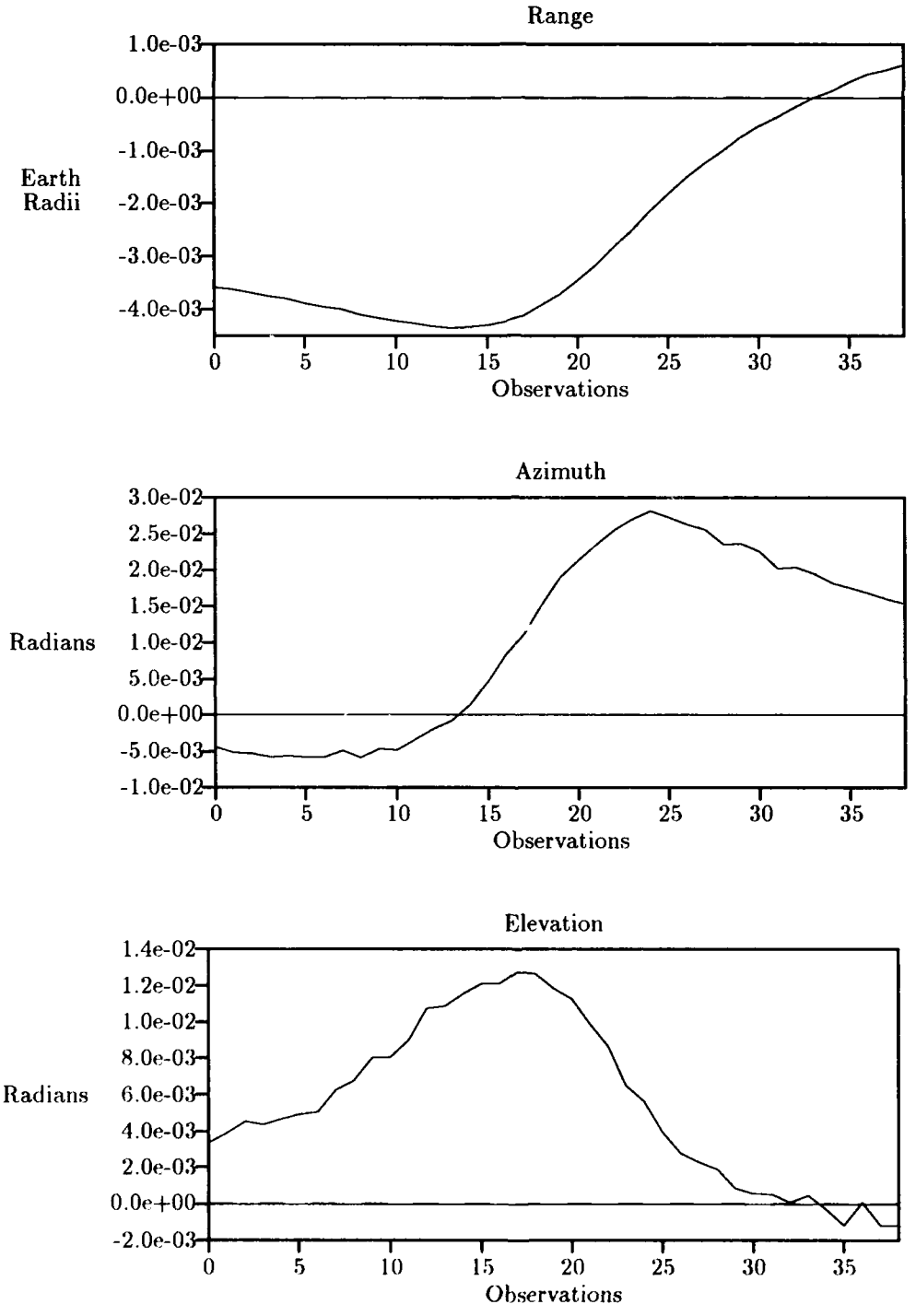


Figure 28. Residual Values Before Differential Correction — Mir

V. Conclusions and Recommendations

5.1 Introduction

The analysis of Chapter 4 shows the equinoctial element, two-body differential corrector is not an adequate estimator for low-earth orbits. The unique parameters in each orbit case test the performance of the corrector in varying situations to identify potential problem areas. Future analysis of data reduction techniques must include additional factors during the model formulation stage. A more complicated model is needed to successfully use an element set and covariance matrix as an input to a new corrector which solves for perturbations.

5.2 Conclusions

The low-orbit cases analyzed exposed the greatest deficiency in the two-body differential corrector. Large J_2 zonal harmonic perturbations cause the estimator to systematically make increasingly large updates to the reference state, maximizing the true error. After a few iterations, the h and k terms become unreasonably large, causing the eccentricity to grow quadratically to values larger than one. For closed earth orbits, this indicates the point of divergence. High-earth orbits have minimal J_2 effects and the estimator meets all standards.

Air drag perturbations in the truth model data are not large enough to inhibit the performance of the estimator. Air drag affects only low-earth orbiting satellites within 100–1000 kilometers and the perturbation is six orders of magnitude smaller than the dominant J_2 term. When the truth model data is based on two-body and air drag effects alone, the estimator converges easily. The higher altitude orbits tested in Cases I–III have no air drag perturbations in this model.

The differential corrector converges independently of orbit altitude. Answering an original research question, the length of the track of data does not affect convergence. With the significant J_2 perturbations removed, the corrector successfully updates the reference trajectory in all five cases with ten data points or less. The correction is still done adequately using the two-body-only

integrator to propagate the reference state vector. This also shows the lack of dependence on the number of observations needed to perform an update.

Tracking data with high maximum elevations caused minor difficulties. Data tracks with high elevations have fast-changing azimuth values causing jumps as high as 40 degrees between observations. The equinoctial equations of motion only change the mean anomaly term between observations and the change is at a constant rate. Therefore, the predicted azimuth values do not rotate as rapidly, resulting in high residuals. This problem is alleviated, somewhat, when the initial integration of the the reference state uses the full integrator. The equinoctial equations of motion work fine for passes with lower maximum elevations due to slower azimuth values change.

Two-step orbit estimation did not solve any of the previous difficulties. The two-step process works by first solving for the in-track elements, ℓ and L, to improve the reference trajectory before solving for all six elements simultaneously. Nevertheless, the correction to ℓ and L are inconsequential when compared to the larger full correction terms. The six element update has a poorly conditioned covariance matrix due to the unobservability of the J_2 perturbation.

Increasing the number of observations by combining several passes does not improve data reduction. As indicated earlier, the number of observations does not hinder the correction process, therefore, combining several passes from the same station only increases the processing time. Combining passes from several stations means that a larger portion of the orbit is described but the corrector still does not converge when there is a significant J_2 perturbation. Larger numbers of observations cause larger initial RMS values resulting in more iterations and larger final RMS values.

5.3 Recommendations

The equinoctial element set two-body least squares estimator is inadequate as a data reduction procedure. The analysis indicates that the model will work if the J_2 perturbation equations are

included during the derivation of the equations of motion and the linearization matrices. Including this perturbation effect when building the differential corrector might allow the process to converge at the lower altitudes where the harmonic term is larger. Adding J_2 equations to the problem formulation eliminates the simplification of working only with two-body equations of motion and greatly complicates the data linearization matrix, H. Finding H in closed form would then be impractical and should now be analytically evaluated through numerical integration techniques.

5.4 Potential Areas of Study

Processing the observation data sequentially using a Kalman filter may eliminate the problems associated with the large J_2 perturbations in low orbits. Least squares is a good starting point for examining these data reduction techniques, but new points of view in estimation theory favor the sequential approach of the Kalman filter. Switching to a new estimation technique may substitute the need to add the J_2 perturbation terms to the equations of motion during the model formulation.

The data reduction techniques studied here produce an element set and covariance matrix without solving for perturbation effects. A follow-on analysis can examine the use of element sets and covariance matrices as inputs to a batch estimation routine that solves for the perturbations. This two-step process should be able to arrive at the same or better solution than from processing the entire set of raw observations in one step.

5.5 Summary

The equinoctial element set greatly simplifies the formulation of the least squares differential corrector. Five out of six equations of motion are trivial to solve, leaving only the mean anomaly term that changes with time. Combining this numerically stable element set with the theory of two-body orbital mechanics simplifies the model formulation and allows for a closed-form solution of the linearization matrices. Yet, this simplification leads to difficulties when trying to correct

orbits with large perturbation terms. The J_2 zonal harmonic perturbation has the largest effect on low-orbiting satellites and causes the estimation process to diverge at altitudes below 1000 kilometers.

Without the influence of the J_2 perturbation affecting the data, the estimation process converged to a solution, successfully reducing the observations to an element set and a covariance matrix. There are non-catastrophic problems with high elevations, and other perturbations, such as air drag, have minimal effects on convergence. Adding the J_2 perturbation terms to the existing model should improve performance to sufficiently handle any orbit type for data reduction. Multiple passes of data provide no significant improvement for convergence and is outside the boundaries of the scope of this investigation.

Data reduction at the tracking sites is a valid solution for handling the increasing number of earth-orbiting objects. Further studies related to this research warrant close examination. These include: adding J_2 to the equinoctial equations of motion, processing data sequentially with a Kalman filter, or trying different element sets. Choosing the right combination of the tools currently available will prove the value of reducing the data to keep an accurate catalog of all orbiting objects.

Appendix A. Equinoctial Elements

The equinoctial element set has significant advantages for the orbit determination problem of earth-orbiting satellites. There are numerical difficulties when the orbit has an eccentricity near zero or an inclination near zero or 90 degrees. When the eccentricity approaches zero, the argument of perigee becomes undefined and as the inclination approaches zero, the node becomes undefined. A canonical transformation from classical elements to the equinoctial elements eliminates the zero eccentricity problem and shifts the inclination difficulties to 180 degrees. This encompasses all problem areas for earth orbits.

Common solution methodology for the two-body problem takes advantage of Hamilton-Jacobi theories. The equations of motion are described by the system Hamiltonian which is based on a set of coordinates, q_i, and momenta, p_i such that

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \qquad \dot{p}_i = -\frac{\partial H}{\partial q_i} \tag{40}$$

where $H(q_i, p_i, t)$ is the Hamiltonian as a function of q_i, p_i, and t. Hamilton proved that there exists a new element set in which all coordinates and momenta are constant making the new Hamiltonian equal to zero. Jacobi showed that any solution to the zero Hamiltonian problem is a solution to the dynamical system (14:2).

Solving the two-body problem in polar coordinates r, θ, and ϕ (see Figure 29) with Hamilton-Jacobi theory yields a set of coordinates and momenta with the desired zero Hamiltonian. Comparing the polar solution to the planar solution yields canonical coordinates and momenta:

Coordinate			Momenta		
$-T_o$	=	$-$perigee passage time	E	=	energy
ω	=	argument of perigee	h	=	angular momentum
Ω	=	node	$h\cos i$	=	z component of h

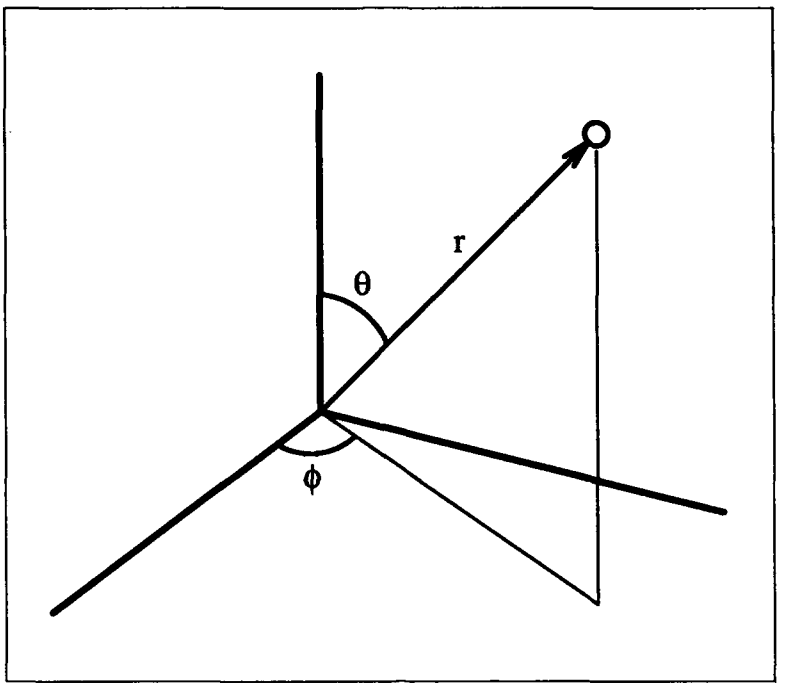


Figure 29. Polar Coordinates for the Two-Body Problem

Because this is a two-body solution only, all six variables are constant. This set of coordinates and momenta suffers difficulties at zero eccentricity and inclinations at zero and ninety degrees (14:17–20).

These canonical coordinates are not the standard set of variables used when solving orbit problems. The *Delaunay elements* contain new quantities for the first coordinate and momenta values and simply rename the other four. The new elements are $\{\ell, g, h, L, G, H\}$ with the coordinates in small letters and the momenta in capital letters. The last four variables are the identity transformations: $g = \omega$, $h = \Omega$, $G = h$, and $H = h \cos i$. Using canonical transformation theory, the complete set of Delaunay elements are:

Coordinate	Momenta
$\ell = M$	$L = \sqrt{\mu a}$
$g = \omega$	$G = L\sqrt{1-e^2}$
$h = \Omega$	$H = G\cos i$

The only time-dependent quantity in this element set is ℓ which makes the new Hamiltonian no longer zero. It is now equal to the total energy, $-\mu/2a = -\mu^2/2L^2$. Because this was simply a renaming of the original element set, there still are problems near zero eccentricity and inclination (14:21).

The problems with the Delaunay elements stem from the use of polar coordinates. By shifting to the rectangular coordinates h and k (see Figure 30), the only remaining problem exists at inclinations near 90 degrees. The transformed Delaunay elements are:

Coordinate	Momenta
$\ell = M$	$L = \sqrt{\mu a}$
$k = e\cos\omega$	$h = e\sin\omega$
$q = \tan i\cos\Omega$	$p = \tan i\sin\Omega$

This set of elements would be fine for non-polar orbits. However, to deal with all Earth orbits, inclination problems near 90 degrees must be eliminated (14:21–22).

The last transformation, to the *equinoctial elements*, is a slight adjustment to the rectangular form of the Delaunay elements. By replacing the $\{\tan i\}$ values with a half-angle $\{\tan(i/2)\}$, the numerical difficulty is shifted to 180 degrees. The elements are now completely defined:

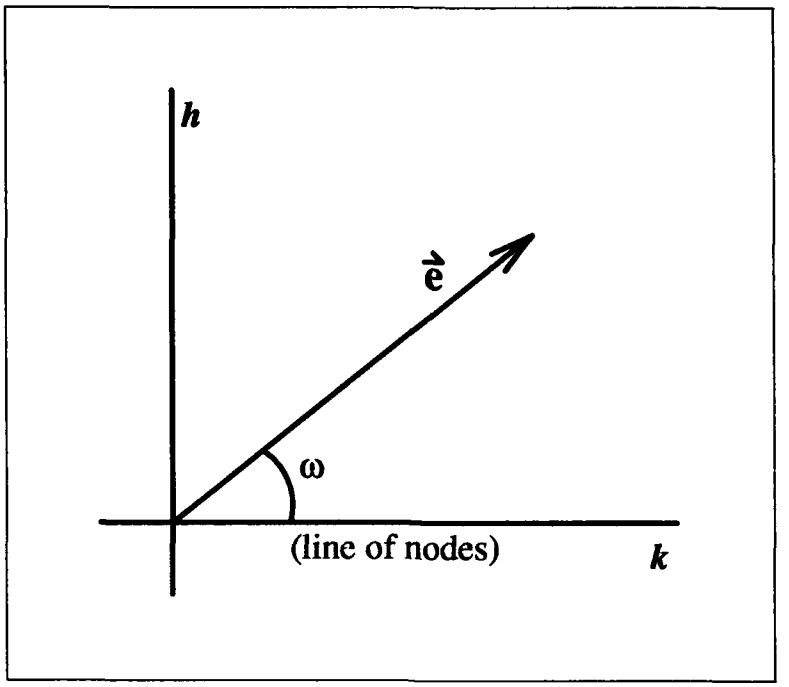


Figure 30. Polar/Rectangular Elements for Delaunay Variables

Coordinate	Momenta
$\ell = M$	$L = \sqrt{\mu a}$
$k = a_f = e\cos\omega$	$h = a_g = e\sin\omega$
$\psi = \tan(i/2)\cos\Omega$	$\chi = \tan(i/2)\sin\Omega$

This is the standard element set for the Space Surveillance Center at NORAD (with the exception of the mean motion n instead of L) (14:22). This element set is also the standard for differential correction at the Consolidated Space Operations Center located at Falcon AFB, CO.

The Hamiltonian for these elements is still a function of the total energy, therefore it is not constant. The only time-dependent value in the equinoctial elements is $\ell = M$, the mean anomaly. For differential correction of the two-body problem, all elements except the mean anomaly remain constant. This greatly simplifies the data linearization matrix, H, shown in Appendix D. Without

the simplifying assumption of working with the two-body problem alone, a closed-form solution is not practical.

Examination of these elements gives an intuitive feel for why the numerical problems have been eliminated. When the eccentricity approaches zero, the argument of perigee became undefined. The new elements h and k are created as a combination of both of these elements so that they are solved simultaneously rather than individually. The undefined node with zero inclination is handled by ψ and χ which combines the problem elements in the same fashion as h and k. The only remaining elements related to the classical set are the mean anomaly, M, and the mean motion term, L, which can each be individually determined without problem (12:6.2-3).

Appendix B. Element Set Transformations

B.1 Earth Centered Inertial (ECI) to Classical Elements

There is a straightforward transformation between a satellite position, $\vec{r}$, and velocity, $\vec{v}$, in geocentric-equatorial coordinates to the classical Keplerian element set $\{a, e, i, \Omega, \omega, M\}$. The first step is to calculate the magnitude of the position and velocity vectors, $r = |\vec{r}|$ and $v = |\vec{v}|$, and then to form three fundamental vectors — $\vec{h}$, $\vec{n}$, and $\vec{e}$. The angular momentum vector is defined as $\vec{h} = \vec{r} \times \vec{v}$ which has components

$$h_1 = r_y v_z - r_z v_y \tag{41}$$

$$h_2 = r_z v_x - r_x v_z \tag{42}$$

$$h_3 = r_x v_y - r_y v_x \tag{43}$$

with $h = |\vec{h}|$ (1:61).

Next, the nodal vector is defined as $\vec{n} = \hat{k} \times \vec{h}$. The individual parts are related to $\vec{h}$ by

$$n_1 = -h_2 \tag{44}$$

$$n_2 = h_1 \tag{45}$$

$$n_3 = 0 \tag{46}$$

with $n = |\vec{n}|$.

The eccentricity vector is somewhat more complicated and is defined as

$$\vec{e} = \frac{1}{\mu}\left[\left(v^2 - \frac{\mu}{r}\right)\vec{r} - (\vec{r} \cdot \vec{v})\vec{v}\right] \tag{47}$$

with $e = |\vec{e}|$ defined as the eccentricity of the orbit and is the first classical element defined. The eccentricity vector points from the center of the Earth to the direction of perigee (1:61–62).

Inclination, the node, and the argument of perigee are easy to find (1:63):

$$i = \cos^{-1}\left(\frac{h_3}{h}\right) \tag{48}$$

$$\Omega = \cos^{-1}\left(\frac{n_1}{n}\right) \tag{49}$$

$$\omega = \cos^{-1}\left(\frac{\vec{n}\cdot\vec{e}}{ne}\right) \tag{50}$$

The last element, the mean anomaly, is more difficult to determine. Both the true anomaly, ν, and the eccentric anomaly, E, must be found first. The relationship of the eccentric anomaly and true anomaly is shown in Figure 31.

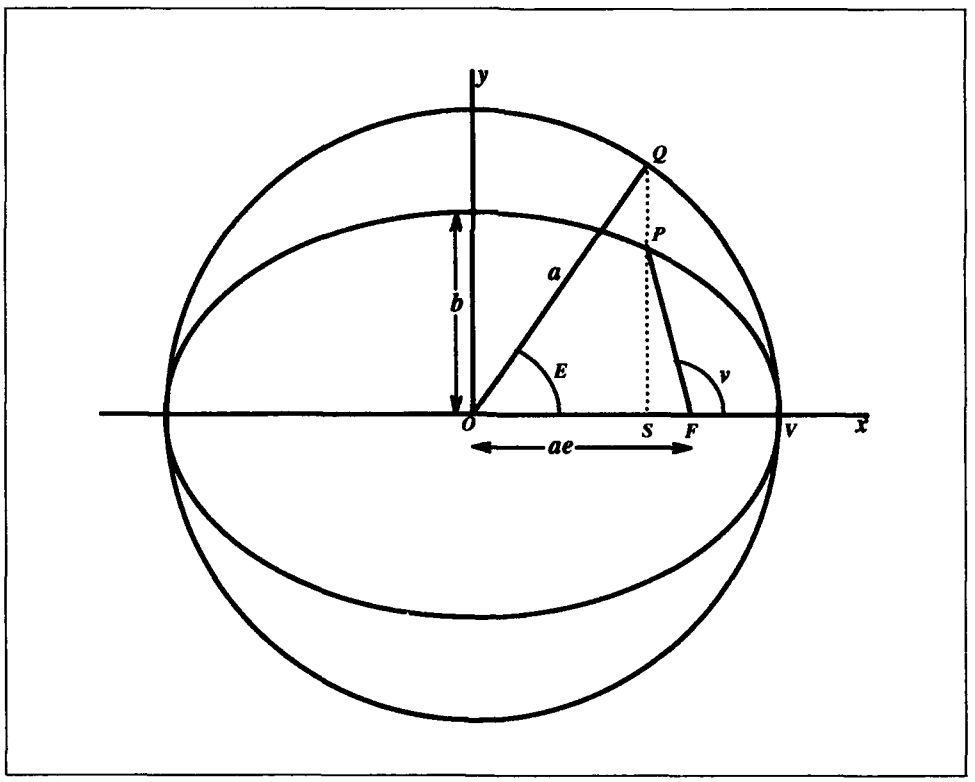


Figure 31. True and Eccentric Anomaly (1:183)

The true anomaly is found in the same fashion as the other classical elements (1:63):

$$\nu = \cos^{-1}\left(\frac{\vec{e}\cdot\vec{r}}{er}\right) \tag{51}$$

From the equation for a conic section and the geometry in Figure 31, a relationship between ν and E can be found based solely on the classical elements. Combining

$$r = \frac{a(1-e^2)}{1+e\cos\nu} \tag{52}$$

and

$$\cos E = \frac{ae + r\cos\nu}{a} \tag{53}$$

gives

$$\cos E = \frac{e+\cos\nu}{1+e\cos\nu} \tag{54}$$

Inverting this relationship gives

$$\cos\nu = \frac{\cos E - e}{1-e\cos E} \tag{55}$$

This relationship and the double-angle identities, $\cos 2A = 1 - 2\sin^2 A$ and $\cos 2A = 2\cos^2 A - 1$, give the half-angle relationships:

$$\sin^2\left(\frac{\nu}{2}\right) = \frac{a(1+e)}{r}\sin^2\left(\frac{E}{2}\right) \tag{56}$$

$$\cos^2\left(\frac{\nu}{2}\right) = \frac{a(1-e)}{r}\cos^2\left(\frac{E}{2}\right) \tag{57}$$

By dividing these two equations and taking the square root, the equation for the tangent follows:

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right) \tag{58}$$

This gives a good relationship between ν and E that does not need to be adjusted for the correct quadrant (2:158–159):

$$E = 2\tan^{-1}\left[\sqrt{\frac{1-e}{1+e}}\tan\left(\frac{\nu}{2}\right)\right] \tag{59}$$

Now that the eccentric anomaly has been found, it is easy to find the mean anomaly. The mean anomaly is the mean angular position of the satellite as measured from the point of perigee. It is defined by Kepler's equation:

$$M = E - e\sin E \tag{60}$$

This completes the conversion from the ECI values to the classical elements (1:185).

B.2 Classical to Equinoctial Elements

The conversion to equinoctial elements is defined by the relationships derived in Appendix A:

$$\begin{aligned} \ell &= M \\ k &= e\cos\omega \\ \psi &= \tan(i/2)\cos\Omega \\ L &= \sqrt{\mu a} \\ h &= e\sin\omega \\ \chi &= \tan(i/2)\sin\Omega \end{aligned} \tag{61}$$

B.3 Equinoctial to Classical Elements

The first step in deriving the classical elements is to perform the reverse transformations of the last section:

$$\begin{aligned} a &= L^2/\mu \\ e &= \sqrt{k^2+h^2} \\ i &= 2\tan^{-1}\sqrt{\psi^2+\chi^2} \\ \Omega &= \sin^{-1}\left(\chi/\sqrt{\psi^2+\chi^2}\right) \\ \omega &= \sin^{-1}\left(h/\sqrt{k^2+h^2}\right) \\ M &= \ell \end{aligned} \tag{62}$$

Both Ω and ω need to be moved into the proper quadrant. The conditions for Ω are

If $\psi < 0$ then	Subtract value from π
If $\psi > 0$ and $\chi < 0$ then	Add 2π
If $\psi > 0$ and $\chi > 0$ then	Value returned is okay

Likewise, the conditions for ω are

If $k < 0$ then	Subtract value from π
If $k > 0$ and $h < 0$ then	Add 2π
If $k > 0$ and $h > 0$ then	Value returned is okay

B.4 Classical to Earth Centered Inertial Elements (ECI)

The classical elements are easily described in the perifocal PQW frame with the orbit plane as the fundamental plane and $\widehat{P}$ pointing in the direction of perigee, $\widehat{Q}$ rotated 90 degrees in the direction of orbital motion, and $\widehat{W}$ perpendicular to the orbit plane along the angular momentum vector (1:57).

First, the mean anomaly, M, must be converted back to the true anomaly through the eccentric anomaly. A Newton iteration scheme based on Kepler's equation, $M = E - e\sin E$, finds the eccentric anomaly (1:222):

Step 1: Choose an initial guess for E_n (like M_o)

Step 2: Calculate $E_{n+1} = E_n + [M_o - (E_n - e \sin E_n)] / (1 - e \cos E_n)$

Step 3: If $|(E_{n+1} - E_n) / E_{n+1}| >$ Tolerance

then set $E_n = E_{n+1}$ and go to Step 2

Step 4: Once below Tolerance, $E = E_{n+1}$

The next pieces needed are quantities specifically relating the classical elements to the perifocal elements (1:187):

$$r = a(1 - e \cos E) \tag{63}$$

$$\cos \nu = \frac{e - \cos E}{e \cos E - 1} \tag{64}$$

$$\sin \nu = \frac{a\sqrt{1 - e^2}}{r} \sin E \tag{65}$$

Now, the vectors in the PQW reference frame can be calculated as

$$\vec{r} = r \cos \nu \hat{P} + r \sin \nu \hat{Q} \tag{66}$$

$$\vec{v} = \sqrt{\frac{\mu}{p}} \left[-\sin \nu \hat{P} + (e + \cos \nu) \hat{Q} \right] \tag{67}$$

which completes the preparation for conversion into the ECI elements (1:72–73).

Converting to ECI elements involves a rotation from the perifocal coordinate system to the Earth centered inertial reference frame. Both the position and velocity terms are rotated by the same matrix, R (1:82–83):

$$R = \begin{bmatrix} \cos\Omega \cos\omega - \sin\Omega \sin\omega \cos i & -\cos\Omega \sin\omega - \sin\Omega \cos\omega \cos i & \sin\Omega \sin i \\ \sin\Omega \cos\omega + \cos\Omega \sin\omega \cos i & -\sin\Omega \sin\omega + \cos\Omega \cos\omega \cos i & -\cos\Omega \sin i \\ \sin\omega \sin i & \cos\omega \sin i & \cos i \end{bmatrix} \tag{68}$$

The transformations are

$$\vec{r}_{ECI} = R\,\vec{r}_{PQW} \tag{69}$$

$$\vec{v}_{ECI} = R\,\vec{v}_{PQW} \tag{70}$$

B.5 Earth Centered Inertial (ECI) to Topocentric Elements

This transformation is especially useful in the differential corrector to determine the predicted range, azimuth, and elevation observations from the updated equinoctial reference trajectory.

B.5.1 Derivation of θ_g. The first step in converting to topocentric observations is to determine the longitude of the Earth station relative to the inertial reference frame. Greenwich mean sidereal time, θ_g, is calculated from the value at the epoch time of 1200 hrs on 1 January 2000.

$$\begin{aligned}
\text{Step 1}: \quad jd &= (t_{ob}/86400.0) + 2440000.0 \\
\text{Step 2}: \quad ut &= \text{mod}(jd + 0.5, 1) \\
\text{Step 3}: \quad jd &= jd - ut \\
\text{Step 4}: \quad tu &= (jd - 2451545.0)/36525.0 \\
\text{Step 5}: \quad gmst &= 24110.54841 + \\
&\quad 8640184.812866\,tu + 0.093104\,tu^2 - (6.2 \times 10^{-6})\,tu^3 \\
\text{Step 6}: \quad gmst &= \text{mod}(gmst + 86400.0\,ut\,(86400.0\,\omega_e/2\pi), 86400.0) \\
\text{Step 7}: \quad \theta_g &= 2\pi(gmst)/86400.0
\end{aligned} \tag{71}$$

First, the Julian day of the observation time, *jd*, is converted back to standard Julian day format (a simplified date, minus 2440000.0, is maintained within the program). The variable *ut* holds the fraction of the day past 0000 hrs. The new *jd* is set to the normal 24 hour clock time. Continuing, *tu* keeps track of the number of days away from 1 January 2000. The Greenwich Mean Sidereal Time, *gmst*, is the desired rotational offset between the rotating topocentric reference frame and

the inertial coordinate system. The main formula handles the offset of total days and then adjusts the value due to the fraction into the day (ut) by multiplying by the rotation of the Earth, ω_e. Finally, θ_g is calculated in radians (4, 8:B6).

B.5.2 Topocentric Elements. With θ_g determined, the longitude at the observation time is

$$\theta = \text{mod}(\theta_g + \lambda, 2\pi) \tag{72}$$

based on the input station longitude, λ (1:100). Next, the input geodetic latitude is converted to geocentric latitude, ϕ, based on the flattening of the reference spheroid, f, and the new site location is found from

$$\begin{aligned} f &= (298.257)^{-1} \\ C &= [1 + (f^2 - 2f)\sin^2\phi]^{-1/2} \\ S &= (1-f)^2 C \end{aligned} \tag{73}$$

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix}_{site} = \begin{bmatrix} (R_e C + h)\cos\phi\cos\theta \\ (R_e C + h)\cos\phi\sin\theta \\ (R_e S + h)\sin\phi \end{bmatrix} \tag{74}$$

which is now in inertial coordinates (5).

The observations are determined by first computing the inertial range vector, $\vec{\rho} = \vec{r} - \vec{R}_{site}$. This is related to the topocentric values by

$$\begin{bmatrix} \rho_s \\ \rho_e \\ \rho_z \end{bmatrix} = \begin{bmatrix} \sin\phi\cos\theta & \sin\phi\sin\theta & -\cos\phi \\ -\sin\theta & \cos\theta & 0 \\ \cos\phi\cos\theta & \cos\phi\sin\theta & \sin\phi \end{bmatrix} \begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} \tag{75}$$

The last step is to determine the observations — the range, azimuth, and elevation associated with this particular position vector and time:

$$\rho = \sqrt{\rho_s^2 + \rho_e^2 + \rho_z^2} \tag{76}$$

$$\text{Az} = \tan^{-1}\left(-\frac{\rho_e}{\rho_s}\right) \tag{77}$$

$$\text{El} = \sin^{-1}\left(\frac{\rho_z}{\rho}\right) \tag{78}$$

The Azimuth value must be shifted into the correct quadrant since the inverse tangent function does not return values for all quadrants (1:79,85).

If $\rho_s > 0$ then	Add π
If $\rho_s < 0$ and $\rho_e < 0$ then	Add 2π
Otherwise	Value returned is okay

Appendix C. Equinoctial Equations of Motion and the State Transition Matrix, Φ

The simplified dynamics using equinoctial elements makes the equations of motion and the state transition matrix easy to solve in closed form. The advantage of using this model appears when propagating the equations of motion for the equinoctial state vector, $X = \{\ell, k, \psi, L, h, \chi\}$. The equations of motion themselves are simplified because only the mean anomaly term, ℓ, changes:

$$\ell = \ell_o + M = \ell_o + n(t - T) = \ell_o + \frac{\mu^2}{L_o^3}\Delta t \tag{79}$$

The remaining equations of motion are

$$\begin{aligned} k &= k_o \\ \psi &= \psi_o \\ L &= L_o \\ h &= h_o \\ \chi &= \chi_o \end{aligned} \tag{80}$$

The state transition matrix (equations of variation) is then found as

$$\Phi = \frac{\partial X(t)}{\partial X(t_o)} = \begin{Bmatrix} 1 & 0 & 0 & -\frac{3\mu^2}{L_o^4}\Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{Bmatrix} \tag{81}$$

These equations of motion and variation are used in the differential corrector when propagating the equinoctial element reference state to determine the predicted observation values.

Appendix D. The Data Linearization Matrix, H

The data linearization matrix, H, is the most math intensive portion of the equinoctial element dynamics model. H is derived by $\partial G/\partial X_{equin}$ where G is the observation matrix consisting of the *observed* range, azimuth, and elevation. X_{equin} is the reference trajectory state vector expressed in the equinoctial elements.

Because this model has been simplified to estimate only two-body effects, H is obtainable in closed form. Extensive use of the chain rule when building H makes the problem easier to maintain. H is broken into three parts:

$$H = \frac{\partial G}{\partial X_{equin}} = \frac{\partial G}{\partial X_{eci}} \frac{\partial X_{eci}}{\partial X_{class}} \frac{\partial X_{class}}{\partial X_{equin}} \tag{82}$$

Each of the parts is individually solved in closed form and then combined through matrix multiplication.

The first relationship, $\partial G/\partial X_{eci}$, relates the observations $G(X)^T = \{\rho_{sez}, Az, El\}$ to the ECI state vector, $X_{eci} = \{r_x, r_y, r_z, v_x, v_y, v_z\}$. These partial derivatives result in a 3 x 6 matrix and are found through

$$\rho_{sez} = \sqrt{\rho_s^2 + \rho_e^2 + \rho_z^2} \tag{83}$$

$$Az = \tan^{-1}\left(-\frac{\rho_e}{\rho_s}\right) \tag{84}$$

$$El = \sin^{-1}\left(\frac{\rho_z}{\rho_{sez}}\right) \tag{85}$$

$$\begin{bmatrix} \rho_s \\ \rho_e \\ \rho_z \end{bmatrix} = \begin{bmatrix} \sin\phi\cos\theta & \sin\phi\sin\theta & -\cos\phi \\ -\sin\theta & \cos\theta & 0 \\ \cos\phi\cos\theta & \cos\phi\sin\theta & \sin\phi \end{bmatrix} \begin{bmatrix} \rho_x \\ \rho_y \\ \rho_z \end{bmatrix} \tag{86}$$

$$\vec{\rho}_{eci} = \{\rho_x, \rho_y, \rho_z\}^T = \vec{r}_{eci} - \vec{R}_{site} \tag{87}$$

Notice that the ECI velocity relationships are not needed as long as range rate is not included in the observation matrix. This makes the right 3×3 part of the $\partial G/\partial X_{eci}$ matrix all zeros.

The resulting partials are then given by

$$
\begin{aligned}
H_1(1,1) &= \partial\rho_{sez}/\partial r_x &&= [\rho_s \sin\phi \cos\theta - \rho_e \sin\theta + \rho_z \cos\phi \cos\theta]/\rho_{sez} \\
H_1(1,2) &= \partial\rho_{sez}/\partial r_y &&= [\rho_s \sin\phi \sin\theta + \rho_e \cos\theta + \rho_z \cos\phi \sin\theta]/\rho_{sez} \\
H_1(1,3) &= \partial\rho_{sez}/\partial r_z &&= [-\rho_s \cos\phi + \rho_z \sin\phi]/\rho_{sez} \\
H_1(2,1) &= \partial Az/\partial r_x &&= [\rho_e \sin\phi \cos\theta + \rho_s \sin\theta]/\left(\rho_s^2 + \rho_e^2\right) \\
H_1(2,2) &= \partial Az/\partial r_y &&= [\rho_e \sin\phi \sin\theta - \rho_s \cos\theta]/\left(\rho_s^2 + \rho_e^2\right) \\
H_1(2,3) &= \partial Az/\partial r_z &&= [-\rho_e \cos\phi]/\left(\rho_s^2 + \rho_e^2\right) \\
H_1(3,1) &= \partial El/\partial r_x &&= [\rho_{sez} \cos\phi \cos\theta - \rho_z H_1(1,1)]/\left(\rho_{sez}\sqrt{\rho_{sez}^2 - \rho_z^2}\right) \\
H_1(3,2) &= \partial El/\partial r_y &&= [\rho_{sez} \cos\phi \sin\theta - \rho_z H_1(1,2)]/\left(\rho_{sez}\sqrt{\rho_{sez}^2 - \rho_z^2}\right) \\
H_1(3,3) &= \partial El/\partial r_z &&= [\rho_{sez} \sin\phi - \rho_z H_1(1,3)]/\left(\rho_{sez}\sqrt{\rho_{sez}^2 - \rho_z^2}\right)
\end{aligned}
\tag{88}
$$

This completes the first 1/3 of the H matrix.

The second partial, $\partial X_{eci}/\partial X_{class}$ relates the ECI state vector, $X_{eci}^T = \{r_x, r_y, r_z, v_x, v_y, v_z\}$ to the classical Keplerian elements, $X_{class} = \{a, e, i, \Omega, \omega, M\}$. These partial derivatives result in a 6×6 matrix. ECI elements are related to the classical elements by

$$
\begin{bmatrix} r_x \\ r_y \\ r_z \\ \cdots \\ v_x \\ v_y \\ v_z \end{bmatrix} = R \begin{bmatrix} r\cos\nu \\ r\sin\nu \\ 0 \\ \cdots \\ -\sqrt{\mu/[a(1-e^2)]}\sin\nu \\ \sqrt{\mu/[a(1-e^2)]}(e+\cos\nu) \\ 0 \end{bmatrix}_{PQW}
\tag{89}
$$

where R is

$$\begin{bmatrix} \cos\Omega\cos\omega - \sin\Omega\sin\omega\cos i & -\cos\Omega\sin\omega - \sin\Omega\cos\omega\cos i & \sin\Omega\sin i \\ \sin\Omega\cos\omega + \cos\Omega\sin\omega\cos i & -\sin\Omega\sin\omega + \cos\Omega\cos\omega\cos i & -\cos\Omega\sin i \\ \sin\omega\sin i & \cos\omega\sin i & \cos i \end{bmatrix} \tag{90}$$

Note that the W elements of position and velocity are zero in the PQW reference frame so the third column of the rotation matrix is unneeded. Now, r and ν are not in the classical element set so expressions relating these variables are

$$r = a(1 - e\cos E) \tag{91}$$

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right) \tag{92}$$

This expression is then solved for ν:

$$\nu = 2\tan^{-1}\left[\sqrt{\frac{1+e}{1-e}}\tan\left(\frac{E}{2}\right)\right] \tag{93}$$

The equations for r and ν each introduced yet another element that is not in the classical element set, the eccentric anomaly, E. Kepler's equation relates E to the classical elements:

$$M = E - e\sin E \tag{94}$$

To determine the 6 x 6 matrix, it is best to begin at the lowest equation (Equation 94) and work to the top. Each partial derivative in the lower equations can be used through the chain rule in follow-on equations.

Partial derivatives from Kepler's equation are solved implicitly:

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial i} = \frac{\partial E}{\partial \Omega} = \frac{\partial E}{\partial \omega} = 0 \tag{95}$$

$$\frac{\partial E}{\partial e} = \frac{\sin E}{1 - e\cos E} \tag{96}$$

$$\frac{\partial E}{\partial M} = \frac{1}{1 - e\cos E} \tag{97}$$

The partial derivatives of the true anomaly are next:

$$\frac{\partial \nu}{\partial a} = \frac{\partial \nu}{\partial i} = \frac{\partial \nu}{\partial \Omega} = \frac{\partial \nu}{\partial \omega} = 0 \tag{98}$$

$$\frac{\partial \nu}{\partial e} = \frac{2}{1 + [(1+e)/(1-e)]\tan^2(E/2)} \left[\frac{1}{2} \frac{\partial E}{\partial e} \sqrt{\frac{1+e}{1-e}} \sec^2\left(\frac{E}{2}\right) + \frac{\tan(E/2)}{(1-e)^2} \sqrt{\frac{1-e}{1+e}} \right] \tag{99}$$

$$\frac{\partial \nu}{\partial M} = (1-e)\sqrt{\frac{1+e}{1-e}} \left(\frac{\partial E}{\partial M}\right)^2 \tag{100}$$

The position vector partial derivatives are

$$\frac{\partial r}{\partial i} = \frac{\partial r}{\partial \Omega} = \frac{\partial r}{\partial \omega} = 0 \tag{101}$$

$$\frac{\partial r}{\partial a} = (1 - c\cos E) \tag{102}$$

$$\frac{\partial r}{\partial e} = ae\sin E \frac{\partial E}{\partial e} - a\cos E \tag{103}$$

$$\frac{\partial r}{\partial M} = ae\sin E \frac{\partial E}{\partial M} \tag{104}$$

Next, it is useful to define the partial derivatives of the elements from the first two columns of the rotation matrix:

$$\begin{aligned}
&\partial R_{11}/\partial a = \partial R_{11}/\partial e = \partial R_{11}/\partial M = 0 && \partial R_{12}/\partial a = \partial R_{12}/\partial e = \partial R_{12}/\partial M = 0 \\
&\partial R_{11}/\partial i = \sin\Omega \sin\omega \sin i && \partial R_{12}/\partial i = \sin\Omega \cos\omega \sin i \\
&\partial R_{11}/\partial \Omega = -R_{21} && \partial R_{12}/\partial \Omega = -R_{22} \\
&\partial R_{11}/\partial \omega = R_{12} && \partial R_{12}/\partial \omega = -R_{11}
\end{aligned} \tag{105}$$

$$\begin{aligned}
&\partial R_{21}/\partial a = \partial R_{21}/\partial e = \partial R_{21}/\partial M = 0 && \partial R_{22}/\partial a = \partial R_{22}/\partial e = \partial R_{22}/\partial M = 0 \\
&\partial R_{21}/\partial i = -\cos\Omega \sin\omega \sin i && \partial R_{22}/\partial i = -\cos\Omega \cos\omega \sin i \\
&\partial R_{21}/\partial \Omega = R_{11} && \partial R_{22}/\partial \Omega = R_{12} \\
&\partial R_{21}/\partial \omega = R_{22} && \partial R_{22}/\partial \omega = -R_{21}
\end{aligned} \tag{106}$$

$$\begin{aligned}
&\partial R_{31}/\partial a = \partial R_{31}/\partial e = \partial R_{31}/\partial M = 0 && \partial R_{32}/\partial a = \partial R_{32}/\partial e = \partial R_{32}/\partial M = 0 \\
&\partial R_{31}/\partial i = \sin\omega \cos i && \partial R_{32}/\partial i = \cos\omega \cos i \\
&\partial R_{31}/\partial \Omega = 0 && \partial R_{32}/\partial \Omega = 0 \\
&\partial R_{31}/\partial \omega = R_{32} && \partial R_{32}/\partial \omega = -R_{31}
\end{aligned} \tag{107}$$

Finally, solve for the values of the second matrix:

$$\begin{aligned}
H_2(1,1) &= \partial r_x/\partial a &&= R_{11}\cos\nu(\partial r/\partial a) + R_{12}\sin\nu(\partial r/\partial a) \\
H_2(1,2) &= \partial r_x/\partial e &&= -rR_{11}\sin\nu(\partial\nu/\partial e) + R_{11}\cos\nu(\partial r/\partial e) \\
&&&\quad + rR_{12}\cos\nu(\partial\nu/\partial e) + R_{12}\sin\nu(\partial r/\partial e) \\
H_2(1,3) &= \partial r_x/\partial i &&= r\cos\nu(\partial R_{11}/\partial i) + r\sin\nu(\partial R_{12}/\partial i) \\
H_2(1,4) &= \partial r_x/\partial \Omega &&= r\cos\nu(\partial R_{11}/\partial \Omega) + r\sin\nu(\partial R_{12}/\partial \Omega) \\
H_2(1,5) &= \partial r_x/\partial \omega &&= r\cos\nu(\partial R_{11}/\partial \omega) + r\sin\nu(\partial R_{12}/\partial \omega) \\
H_2(1,6) &= \partial r_x/\partial M &&= -rR_{11}\sin\nu(\partial\nu/\partial M) + R_{11}\cos\nu(\partial r/\partial M) \\
&&&\quad + rR_{12}\cos\nu(\partial\nu/\partial M) + R_{12}\sin\nu(\partial r/\partial M)
\end{aligned} \tag{108}$$

$$
\begin{aligned}
H_2(2,1) &= \partial r_y/\partial a &&= R_{21}\cos\nu(\partial r/\partial a) + R_{22}\sin\nu(\partial r/\partial a) \\
H_2(2,2) &= \partial r_y/\partial e &&= -rR_{21}\sin\nu(\partial\nu/\partial e) + R_{21}\cos\nu(\partial r/\partial e) \\
&&&\quad + rR_{22}\cos\nu(\partial\nu/\partial e) + R_{22}\sin\nu(\partial r/\partial e) \\
H_2(2,3) &= \partial r_y/\partial i &&= r\cos\nu(\partial R_{21}/\partial i) + r\sin\nu(\partial R_{22}/\partial i) \\
H_2(2,4) &= \partial r_y/\partial\Omega &&= r\cos\nu(\partial R_{21}/\partial\Omega) + r\sin\nu(\partial R_{22}/\partial\Omega) \\
H_2(2,5) &= \partial r_y/\partial\omega &&= r\cos\nu(\partial R_{21}/\partial\omega) + r\sin\nu(\partial R_{22}/\partial\omega) \\
H_2(2,6) &= \partial r_y/\partial M &&= -rR_{21}\sin\nu(\partial\nu/\partial M) + R_{21}\cos\nu(\partial r/\partial M) \\
&&&\quad + rR_{22}\cos\nu(\partial\nu/\partial M) + R_{22}\sin\nu(\partial r/\partial M)
\end{aligned}
\tag{109}
$$

$$
\begin{aligned}
H_2(3,1) &= \partial r_z/\partial a &&= R_{31}\cos\nu(\partial r/\partial a) + R_{32}\sin\nu(\partial r/\partial a) \\
H_2(3,2) &= \partial r_z/\partial e &&= -rR_{31}\sin\nu(\partial\nu/\partial e) + R_{31}\cos\nu(\partial r/\partial e) \\
&&&\quad + rR_{32}\cos\nu(\partial\nu/\partial e) + R_{32}\sin\nu(\partial r/\partial e) \\
H_2(3,3) &= \partial r_z/\partial i &&= r\cos\nu(\partial R_{31}/\partial i) + r\sin\nu(\partial R_{32}/\partial i) \\
H_2(3,4) &= \partial r_z/\partial\Omega &&= 0 \\
H_2(3,5) &= \partial r_z/\partial\omega &&= r\cos\nu(\partial R_{31}/\partial\omega) + r\sin\nu(\partial R_{32}/\partial\omega) \\
H_2(3,6) &= \partial r_z/\partial M &&= -rR_{31}\sin\nu(\partial\nu/\partial M) + R_{31}\cos\nu(\partial r/\partial M) \\
&&&\quad + rR_{32}\cos\nu(\partial\nu/\partial M) + R_{32}\sin\nu(\partial r/\partial M)
\end{aligned}
\tag{110}
$$

To simplify the remaining equations, it is convenient to define an intermediate variable:

$$
\mathcal{S} = \sqrt{\frac{\mu}{a(1-e^2)}} \tag{111}
$$

$$
\begin{aligned}
H_2(4,1) &= \partial v_x/\partial a &&= R_{11}\mathcal{S}(\sin\nu/2a) - R_{12}\mathcal{S}(e+\cos\nu)/2a \\
H_2(4,2) &= \partial v_x/\partial e &&= -R_{11}\mathcal{S}\cos\nu(\partial\nu/\partial e) - R_{11}\mathcal{S}(e\sin\nu)/(1-e^2) \\
&&&\quad +R_{12}\mathcal{S}[1-\sin\nu(\partial\nu/\partial e)] + R_{12}\mathcal{S}[e(e+\cos\nu)]/(1-e^2) \\
H_2(4,3) &= \partial v_x/\partial i &&= -\mathcal{S}\sin\nu(\partial R_{11}/\partial i) + \mathcal{S}(e+\cos\nu)(\partial R_{12}/\partial i) \\
H_2(4,4) &= \partial v_x/\partial\Omega &&= -\mathcal{S}\sin\nu(\partial R_{11}/\partial\Omega) + \mathcal{S}(e+\cos\nu)(\partial R_{12}/\partial\Omega) \\
H_2(4,5) &= \partial v_x/\partial\omega &&= -\mathcal{S}\sin\nu(\partial R_{11}/\partial\omega) + \mathcal{S}(e+\cos\nu)(\partial R_{12}/\partial\omega) \\
H_2(4,6) &= \partial v_x/\partial M &&= -R_{11}\mathcal{S}\cos\nu(\partial\nu/\partial M) - R_{12}\mathcal{S}\sin\nu(\partial\nu/\partial M)
\end{aligned}
\tag{112}
$$

$$
\begin{aligned}
H_2(5,1) &= \partial v_y/\partial a &&= R_{21}\mathcal{S}(\sin\nu/2a) - R_{22}\mathcal{S}(e+\cos\nu)/2a \\
H_2(5,2) &= \partial v_y/\partial e &&= -R_{21}\mathcal{S}\cos\nu(\partial\nu/\partial e) - R_{21}\mathcal{S}(e\sin\nu)/(1-e^2) \\
&&&\quad +R_{22}\mathcal{S}[1-\sin\nu(\partial\nu/\partial e)] + R_{22}\mathcal{S}[e(e+\cos\nu)]/(1-e^2) \\
H_2(5,3) &= \partial v_y/\partial i &&= -\mathcal{S}\sin\nu(\partial R_{21}/\partial i) + \mathcal{S}(e+\cos\nu)(\partial R_{22}/\partial i) \\
H_2(5,4) &= \partial v_y/\partial\Omega &&= -\mathcal{S}\sin\nu(\partial R_{21}/\partial\Omega) + \mathcal{S}(e+\cos\nu)(\partial R_{22}/\partial\Omega) \\
H_2(5,5) &= \partial v_y/\partial\omega &&= -\mathcal{S}\sin\nu(\partial R_{21}/\partial\omega + \mathcal{S}(e+\cos\nu)(\partial R_{22}/\partial\omega) \\
H_2(5,6) &= \partial v_y/\partial M &&= -R_{21}\mathcal{S}\cos\nu(\partial\nu/\partial M) - R_{22}\mathcal{S}\sin\nu(\partial\nu/\partial M)
\end{aligned}
\tag{113}
$$

$$
\begin{aligned}
H_2(6,1) &= \partial v_z/\partial a &&= R_{31}\mathcal{S}(\sin\nu/2a) - R_{32}\mathcal{S}(e+\cos\nu)/2a \\
H_2(6,2) &= \partial v_z/\partial e &&= -R_{31}\mathcal{S}\cos\nu(\partial\nu/\partial e) - R_{31}\mathcal{S}(e\sin\nu)/(1-e^2) \\
&&&\quad +R_{32}\mathcal{S}[1-\sin\nu(\partial\nu/\partial e)] + R_{32}\mathcal{S}[e(e+\cos\nu)]/(1-e^2) \\
H_2(6,3) &= \partial v_z/\partial i &&= -\mathcal{S}\sin\nu(\partial R_{31}/\partial i) + \mathcal{S}(e+\cos\nu)(\partial R_{32}/\partial i) \\
H_2(6,4) &= \partial v_z/\partial\Omega &&= 0 \\
H_2(6,5) &= \partial v_z/\partial\omega &&= -\mathcal{S}\sin\nu(\partial R_{31}/\partial\omega) + \mathcal{S}(e+\cos\nu)(\partial R_{32}/\partial\omega) \\
H_2(6,6) &= \partial v_z/\partial M &&= -R_{31}\mathcal{S}\cos\nu(\partial\nu/\partial M) - R_{32}\mathcal{S}\sin\nu(\partial\nu/\partial M)
\end{aligned}
\tag{114}
$$

$\partial X_{class}/\partial X_{equin}$ completes the building of the H matrix. This 6×6 matrix is the easiest of the three to construct and is based on previously defined relationships:

$$\begin{aligned} a &= L^2/\mu \\ e &= \sqrt{k^2+h^2} \\ i &= 2\tan^{-1}\sqrt{\psi^2+\chi^2} \\ \Omega &= \sin^{-1}\left(\chi/\sqrt{\psi^2+\chi^2}\right) \\ \omega &= \sin^{-1}\left(h/\sqrt{k^2+h^2}\right) \\ M &= \ell \end{aligned} \tag{115}$$

The partials are

$$\begin{aligned} H_3(1,4): &\quad \partial a/\partial L = 2L/\mu \\ H_3(2,2): &\quad \partial e/\partial k = k/\sqrt{k^2+h^2} \\ H_3(2,5): &\quad \partial e/\partial h = h/\sqrt{k^2+h^2} \\ H_3(3,3): &\quad \partial i/\partial \psi = 2\psi/\left[(1+\psi^2+\chi^2)\sqrt{\psi^2+\chi^2}\right] \\ H_3(3,6): &\quad \partial i/\partial \chi = 2\chi/\left[(1+\psi^2+\chi^2)\sqrt{\psi^2+\chi^2}\right] \\ H_3(4,3): &\quad \partial \Omega/\partial \psi = -\chi/(\psi^2+\chi^2) \\ H_3(4,6): &\quad \partial \Omega/\partial \chi = \psi/(\psi^2+\chi^2) \\ H_3(5,2): &\quad \partial \omega/\partial k = -h/(k^2+h^2) \\ H_3(5,5): &\quad \partial \omega/\partial h = k/(k^2+h^2) \\ H_3(6,1): &\quad \partial M/\partial \ell = 1 \end{aligned} \tag{116}$$

All other partial derivatives for this matrix are zero.

H is now constructed by matrix multiplication of each of the three parts outlined above. This rigorous approach using partial derivatives is exact (dynamically) and there is no programming need to show H in its final form. Use of the FORTRAN programming language to construct H is shown in Appendix F ('obser.for').

Appendix E. Truth Model Program Listing

E.1 The Truth Model

The truth model is the data generating program with subroutines used to create the observation values range, azimuth, and elevation for each of nine remote tracking stations. Output units are kilometers, seconds, and degrees. The subroutines 'haming,' 'julday,' and 'caldat' and the functions 'rand' and 'randg' were written by Dr. Wiesel.

```
      program truth
c
      common /ham/ t,y(42,4),f(42,4),err(42),dt,mode,n
      double precision t,y,f,err,dt
      integer mode,n
c
      double precision to,tf,ct,dt
      integer iyr,imon,iday,ihr,imin,nstep,iy
      real sec
c
      common /elmnt/ per
      double precision per
c
c     iy is the seed value for the random number generator
c
      open(31,FILE='all.dat')
      iy = 851501
c
c     read initial state vector and place in y
c
      call open
c
c     read in initial time = to
c
      read (3,*) iyr,imon,iday,ihr,imin,sec
      call julday(iyr,imon,iday,ihr,imin,sec,to)
      write (*,9020) iyr,imon,iday,ihr,imin,sec
      write (4,9020) iyr,imon,iday,ihr,imin,sec
c
c     read in final time = tf
c
      read (3,*) iyr,imon,iday,ihr,imin,sec
      call julday(iyr,imon,iday,ihr,imin,sec,tf)
c
c     read in the stepsize = dt (in seconds)
c
      read (3,*) dt
c
c     read in initial r and v
c
      read (3,*) (y(ii,1),ii=1,3)
      read (3,*) (y(ii,1),ii=4,6)
c
c     write initial point to output file = output.t
c
      write (4,9010) (y(ii,1), ii=1,6)
c
c     convert to,tf into seconds for haming routine
```

```
c
      to = to * 86400.0d+0
      tf = tf * 86400.0d+0
c
c     initialize haming
c          mode = 0 means no phi matrix, eom only (n = 6 elements)
c          dt = # points in an orbital period
c
      call elemnt(1)
      t = to
      mode = 0
      n = 6
      nstep = (tf-to)/dt
c
c     have to worry about possible truncation in calculation of nstep
c
      if (dabs(to + dble(nstep)*dt -tf) .gt.
     1    dabs(to + dble(nstep)*dt + dt - tf)) nstep = nstep + 1
c
      nxt = 0
      call haming(nxt)
      if (nxt .eq. 0) go to 100
c
c     write out every 10th position and velocity data point to 'output.t'
c     output of the range, azimuth, and elevation are in 'rts.for'
c
      icount = 0
      call rts(1,iy)
      do 20 i = 1,nstep
           call haming(nxt)
           call rts(nxt,iy)
           if (mod(i,10) .eq. 0) then
                ct = t/86400.0d+0
                call caldat(ct,iyr,imon,iday,ihr,imin,sec)
                write (4,9020) iyr,imon,iday,ihr,imin,sec
                write (4,9010) (y(ii,nxt), ii=1,6)
           endif
  20  continue
c
c     check the last data pt against the first
c
      ct = t/86400.0d+0
      call caldat(ct,iyr,imon,iday,ihr,imin,sec)
      write (*,9020) iyr,imon,iday,ihr,imin,sec
      call elemnt(nxt)
c
 100  call close
      close(31)
c
c
 9010 format(2x,3(f13.6,1x),3(f9.6,1x),/)
 9020 format(3x,i4,1x,4(i2,1x),f5.2)
c
c
      end
c
      include 'open.for'
      include 'close.for'
      include 'haming.for'
      include 'julian.for'
      include 'elemnt.for'
      include 'rts.for'
```

```
*******************************************************************************
      subroutine haming(nxt)
c
c     haming is an ordinary differential equations integrator
c     it is a fourth order predictor-corrector algorithm which means
c     that it carries along the last four values of the state
c     vector, and extrapolates these values to obtain the next
c     value (the prediction part) and then corrects the extrapolated
c     value to find a new value for the state vector.
c
c     the value nxt in the call specifies which of the 4 values
c     of the state vector is the "next" one.
c     nxt is updated by haming automatically, and is zero on
c     the first call
c
c     the user supplies an external routine rhs(nxt) which
c     evaluates the equations of motion
c
      common /ham/ t,y(42,4),f(42,4),errest(42),dt,mode,n
      double precision t,y,f,errest,dt
      integer mode,n
      double precision dt2,to
c
c     all of the good stuff is in this common block.
c     t is the independent variable ( time )
c     y(6,4) is the state vector- 4 copies of it, with nxt
c                 pointing at the next one
c     f(6,4) are the equations of motion, again four copies
c                 a call to rhs(nxt) updates an entry in f
c     errest is an estimate of the truncation error - normally not used
c     n is the number of equations being integrated - 6 or 42 here
c     h is the time step
c     mode is 0 for just EOM, 1 for both EOM and EOV
c
      tol = 0.0000000001d+00
c     switch on starting algorithm or normal propagation
      if(nxt) 190,10,200
c
c     this is hamings starting algorithm....a predictor - corrector
c     needs 4 values of the state vector, and you only have one- the
c     initial conditions.
c     haming uses a Picard iteration (slow and painfull) to get the
c     other three.
c     if it fails, nxt will still be zero upon exit, otherwise
c     nxt will be 1, and you are all set to go
c
   10 to = t
      dt2 = dt/2.0d+00
      call rhs(1)
      do 40 l = 2,4
      t = t + dt2
      do 20 i = 1,n
   20 y(i,l) = y(i,l-1) + dt2*f(i,l-1)
      call rhs(l)
      t = t + dt2
      do 30 i = 1,n
   30 y(i,l) = y(i,l-1) + dt*f(i,l)
   40 call rhs(l)
      jsw = -10
   50 isw = 1
      do 120 i = 1,n
      dt2 = y(i,1) + dt*( 9.0d+00*f(i,1) + 19.0d+00*f(i,2)
     1    - 5.0d+00*f(i,3) + f(i,4) ) / 24.0d+00
      if( dabs( dt2 - y(i,2)) .lt. tol ) go to 70
      isw = 0
```

```
   70 y(i,2) = dt2
      dt2 = y(i,1) + dt*( f(i,1) + 4.0d+00*f(i,2) + f(i,3))/3.0d+00
      if( dabs( dt2-y(i,3)) .lt. tol ) go to 90
      isw = 0
   90 y(i,3) = dt2
      dt2 = y(i,1) + dt*( 3.0d+00*f(i,1) + 9.0d+00*f(i,2)
     1     + 9.0d+00*f(i,3) + 3.0d+00*f(i,4) ) / 8.0d+00
      if( dabs(dt2-y(i,4)) .lt. tol ) go to 110
      isw = 0
  110 y(i,4) = dt2
  120 continue
      t = to
      do 130 l = 2,4
      t = t + dt
  130 call rhs(l)
      if(isw) 140,140,150
  140 jsw = jsw + 1
      if(jsw) 50,280,280
  150 t = to
      isw = 1
      jsw = 1
      do 160 i = 1,n
  160 errest(i) = 0.0d+0
      nxt = 1
      go to 280
  190 jsw = 2
      nxt = iabs(nxt)
c
c     this is hamings normal propagation loop
c
  200 t = t + dt
      np1 = mod(nxt,4) + 1
      go to (210,230),isw
c     permute the index nxt modulo 4
  210 go to (270,270,270,220),nxt
  220 isw = 2
  230 nm2 = mod(np1,4) + 1
      nm1 = mod(nm2,4) + 1
      npo = mod(nm1,4) + 1
c
c     this is the predictor part
c
      do 240 i = 1,n
      f(i,nm2) = y(i,np1) + 4.0d+00*dt*( 2.0d+00*f(i,npo) - f(i,nm1)
     1    + 2.0d+00*f(i,nm2) ) / 3.0d+00
      y(i,np1) = f(i,nm2) - 0.925619835*errest(i)
  240 continue
c
c     now the corrector - fix up the extrapolated state
c     based on the better value of the equations of motion
c
      call rhs(np1)
      do 250 i = 1,n
      y(i,np1) = ( 9.0d+00*y(i,npo) - y(i,nm2) + 3.0d+00*dt*( f(i,np1)
     1    + 2.0d+00*f(i,npo) - f(i,nm1) ) ) / 8.0d+00
      errest(i) = f(i,nm2) - y(i,np1)
      y(i,np1) = y(i,np1) + 0.0743801653 * errest(i)
  250 continue
      go to (260,270),jsw
  260 call rhs(np1)
  270 nxt = np1
  280 return
      end
c
      include 'rhs.for'
```

```
*******************************************************************************

      subroutine rhs(nxt)
c
c     rhs calculates the equations of motion for an earth orbiting satellite
c
c     the state vector is stored as:
c         y(1-3,nxt) are the x, y, z components of the position vector
c         y(4-6,nxt) are the x, y, z components of the velocity vector
c
c     hamming common
c
      common /ham/ t,y(42,4),f(42,4),err(42),dt,mode,n
      double precision t,y,f,err,dt
      integer mode,n
c
c     general double precision statements
c
      double precision r,vmag,v(3),rcube,cj,cz
      double precision emu,ejtwo,rearth,wearth
      double precision rho,bstar,drag
c
c     the basic function of rhs is to calculate the equations of
c     motion (the f entries) from the given current state (stored in y)
c     and the time t
c
c     earth emu value for units of km, sec
c
      emu    = 3.986012d+5
      ejtwo  = 0.0010827d+0
      rearth = 6378.137d+0
      wearth = 7.292115856d-5
      bstar  = 2.0d+0 * 7.5d+0 / 1000.0d+0
c
c     ************************************
c
c       EVALUATE THE EQUATIONS OF MOTION
c
c     ************************************
c
c     position dot = velocity vector
c
      f(1,nxt) = y(4,nxt)
      f(2,nxt) = y(5,nxt)
      f(3,nxt) = y(6,nxt)
c
c     velocity dot = -two-body - J2 - air drag
c
c     two-body terms first
c
      r = dsqrt(y(1,nxt)*y(1,nxt)+y(2,nxt)*y(2,nxt)+y(3,nxt)*y(3,nxt))
      rcube = r**3
      f(4,nxt) = -emu * y(1,nxt) / rcube
      f(5,nxt) = -emu * y(2,nxt) / rcube
      f(6,nxt) = -emu * y(3,nxt) / rcube
c
c     now the J2 terms
c
      cj = 1.5d+0 * ejtwo * (rearth/r) * (rearth/r)
      cz = 5.0d+0 * (y(3,nxt)/r) * (y(3,nxt)/r)
      f(4,nxt) = f(4,nxt) * (1.0d+0 + cj * (1.0d+0 - cz))
      f(5,nxt) = y(2,nxt)*f(4,nxt)/y(1,nxt)
      f(6,nxt) = f(6,nxt) * (1.0d+0 + cj * (3.0d+0 - cz))
c
```

```
c     now the air drag terms
c
c     bstar = C_d*A/m = 2*7.5/1000 = m^2/kg
c     rho = kg/m^3
c     v = m/sec
c
      call densty(r,rho)
      v(1) = f(1,nxt) + wearth*y(2,nxt)
      v(2) = f(2,nxt) - wearth*y(1,nxt)
      v(3) = f(3,nxt)
      vmag = dsqrt(v(1)**2 + v(2)**2 + v(3)**2)
      drag = 0.5d+0 * bstar * rho * vmag
      f(4,nxt) = f(4,nxt) - drag * v(1)
      f(5,nxt) = f(5,nxt) - drag * v(2)
      f(6,nxt) = f(6,nxt) - drag * v(3)
c
      return
      end
c
      include 'density.for'

*******************************************************************************

      subroutine densty(r,rho)
c
c     density calculates the atmospheric density for altitude 100-1000km
c          density values are based on the 1976 U.S. Standard Atmosphere
c          units are kg/m**3
c
      double precision r,rho,alt,z(9)
c
c     equation constants
c
      z(1) = 246482.873d+0
      z(2) = 537132.030d+0
      z(3) = 1536228.60d+0
      z(4) = 19174788.0d+0
      z(5) = 105.381650d+0
      z(6) = 4968.56120d+0
      z(7) = 56060.0780d+0
      z(8) = 160314.760d+0
      z(9) = 344944.780d+0
c
c     altitude is in meters
c
      alt = (r - 6378.137d+0)*1000.0d+0
      if (alt .lt. 100000.0d+0) then
           rho = 1.0d+50
      elseif (alt .ge. 100000.0d+0 .and. alt .le. 1000000.0d+0) then
           rho =  dexp( -(z(1)/alt)**4 + (z(2)/alt)**3 - (z(3)/alt)**2
     1                + (z(4)/alt)    - (alt/z(9))**4 + (alt/z(8))**3
     2                - (alt/z(7))**2 + (alt/z(6))    - z(5) )
      else
           rho = 0.0d+0
      endif
c
      icount = icount + 1
      if (icount .eq. 1) then
           write(*,9000) alt/1000d+0
      endif
 9000 format (2x,'alt',2x,f19.13,3x,'km',/)
c
      return
      end
```

```
*******************************************************************************

      subroutine rts(nxt,iy)
c
c     This routine controls the output of range, azimuth, and elevation
c     observations to files associated with each input station.
c     Station information is read in as geodetic latitude(phi) and East
c     longitude(lon) in degrees (converted to radians), and altitude in km.
c
      common /ham/ t,y(42,4),f(42,4),err(42),dt,mode,n
      double precision t,y,f,err,dt
      integer mode,n
c
      double precision rsite(3),rho(6)
      double precision pi,degrad,we,flat,rearth
      double precision ut,jd,tu,gmst,thetag
      double precision phi,lon,h,theta,c,s,ct
      double precision range,az,el
      integer isite
      character*4,site
c
      integer iy
      real randg,sigma(3)
      external randg
c
c     constants
c
      pi = 3.141592653589d+0
      degrad = 57.2957795131d+0
      we = 7.292115856d-5
      flat = 1.0d+0/298.257d+0
      rearth = 6378.137d+0
c
c     observation sigmas - range(km), azimuth(deg), elevation(deg)
c
      sigma(1) = 0.1
      sigma(2) = 0.025
      sigma(3) = 0.025
c
c     read in site identifier, phi and lon in degrees (immediately
c     converted to radians), and altitude in meters
c
      open(15,FILE='sensor.loc')
c
c     Reference: 1992 Astronomical Almanac, page B6
c
      jd = t/86400.0d+0 + 2440000.0d+0
      ut = dmod(jd + 0.5d+0,1)
      jd = jd - ut
      tu = (jd-2451545.0d+0)/36525.0d+0
      gmst = 24110.54841d+0 + tu*(8640184.812866d+0 + tu*(0.093104d+0 -
     #       tu*6.2d-6))
      gmst = dmod(gmst + 86400.0d+0*(86400.0d+0*we/(2.0d+0*pi))*ut,
     #            86400.0d+0)
      thetag = 2.0d+0*pi*gmst/86400.0d+0
c
c     control loop for each of nine stations
c          1 = Indi  4 = Hula  7 = Boss
c          2 = Reef  5 = Cook  8 = Pogo
c          3 = Guam  6 = Pike  9 = Lion
c
      do 100 i = 1,9
           read (15,*) site,isite,phi,lon,h
           phi = phi/degrad
           lon = lon/degrad
```

```
          h = h/1000.0d+0
c
c         Convert longitude and time to Local Sidereal Time
c         Reference:  BM & W page 100
c
          theta = dmod(lon + thetag,2.0d+0*pi)
c
c         Convert Geodetic Latitude to Geocentric Latitude
c         Reference:  1992 Astronautical Almanac  page K11
c
          c = 1.0d+0/dsqrt(1.0d+0 + flat*(flat - 2.0d+0)*dsin(phi)**2)
          s = (1.0d+0 - flat)**2 * c
c
c         compute site position and geocentric latitude
c
          rsite(1) = (rearth*c + h)*dcos(phi)*dcos(theta)
          rsite(2) = (rearth*c + h)*dcos(phi)*dsin(theta)
          rsite(3) = (rearth*s + h)*dsin(phi)
c
c         rho(1-3) are the components of range - inertial
c         rho(4-6) are topocentric
c         Reference:  BM & W page 79
c
          do 10 l = 1,3
               rho(l) = y(l,nxt) - rsite(l)
 10       continue
c
          rho(4) = dsin(phi)*dcos(theta)*rho(1) +
     #             dsin(phi)*dsin(theta)*rho(2) -
     #             dcos(phi)*rho(3)
          rho(5) = -dsin(theta)*rho(1) + dcos(theta)*rho(2)
          rho(6) = dcos(phi)*dcos(theta)*rho(1) +
     #             dcos(phi)*dsin(theta)*rho(2) +
     #             dsin(phi)*rho(3)
c
c         calculate the range, azimuth, and elevation
c         Reference:  BM & W page 85
c
          range = dsqrt(rho(4)**2 + rho(5)**2 + rho(6)**2)
          range = range + randg(iy)*sigma(1)
c
          if (rho(4) .gt. 0.0d+0) then
               az = 180.0d+0 + datan(-rho(5)/rho(4)) * degrad
          elseif (rho(5) .lt. 0.0d+0) then
               az = 360.0d+0 + datan(-rho(5)/rho(4)) * degrad
          else
               az = datan(-rho(5)/rho(4)) * degrad
          endif
          az = az + randg(iy)*sigma(2)
c
          el = dasin(rho(6)/range) * degrad
          el = el + randg(iy)*sigma(3)
c
          if (el .gt. 0.0d+0) then
               ct = t/86400.0d+0
               call caldat(ct,iyr,imon,iday,ihr,imin,sec)
               write(i+15,900) iyr,imon,iday,ihr,imin,sec,i,range,az,el
               write(31,900) iyr,imon,iday,ihr,imin,sec,i,range,az,el
          endif
c
 100  continue
c
 900  format(1x,i4,1x,4(i2,1x),f5.2,2x,i2,2x,f12.6,2x,f11.6,2x,f11.6)
c
      close(15)
```

```
      return
      end


      function rand(iy)
c
c     pseudo random number generator on interval (0,1)
c     Collected Algorithms of the CACM #266
c     assumes 2**31 integer math
c     iy is odd integer between 0 and 67108863
c     don't change it after first call
c
      integer k(3),iy
      data k/25,25,5/
c
      do 10 i = 1,3
           iy = k(i)*iy
           iy = iy - (iy/67108864)*67108864
 10   continue
c
      rand = real(iy)/67108864.0
c
      return
      end


      function randg(iy)
c
c     gaussian pseudo random number generator
c     unit variance, zero mean
c     uses central limit theorem with 10 iterates,
c     empirical constant for sigma
c
      real rand
      external rand
      integer iy
c
      r = 0.0
      do 50 i = 1,10
           r = r + rand(iy)
 50   continue
c
      randg = 1.0875*(r - 5.0)
c
      return
      end

*******************************************************************************

      subroutine elemnt(i)
c
c     calculate orbital elements in kilometers and degrees
c
c     haming common block
c
      common /ham/ t,y(42,4),f(42,4),err(42),dt,mode,n
      double precision t,y,f,err,dt
      integer mode,n
c
c     orbital element common block
c
      common /elmnt/ per
      double precision per
c
c     general double precision statements
```

```
c
      double precision axis,ecc,incl,node,pgee,true,eanom,mean
      double precision emu,pi,r,v,h(3),hmag,vn(3),vnmag
      double precision rdotv,f1,f2,e(3),p
      double precision edotn,en,permin,degrad,edotr,er
c
c     constants
c
      emu = 3.986012d+5
      pi = 3.1415926535889d+0
      degrad = 57.2957795131d+0
c
      r = dsqrt(y(1,i)*y(1,i) + y(2,i)*y(2,i) + y(3,i)*y(3,i))
      v = dsqrt(y(4,i)*y(4,i) + y(5,i)*y(5,i) + y(6,i)*y(6,i))
c
c     angular momentum vector, h = r x v, and magnitude, hmag
c
      h(1) = y(2,i)*y(6,i) - y(3,i)*y(5,i)
      h(2) = y(3,i)*y(4,i) - y(1,i)*y(6,i)
      h(3) = y(1,i)*y(5,i) - y(2,i)*y(4,i)
      hmag = dsqrt(h(1)*h(1) + h(2)*h(2) + h(3)*h(3))
c
c     nodal vector, vn = k x h, and magnitude, vnmag
c
      vn(1) = -h(2)
      vn(2) = h(1)
      vn(3) = 0.0d+0
      vnmag = dsqrt(vn(1)*vn(1) + vn(2)*vn(2) + vn(3)*vn(3))
c
c     inclination = incl
c
      incl = dacos(h(3)/hmag)
c
c     right ascension of ascending node = node
c
      if (vnmag .eq. 0.0d+0) then
           print *,'Equatorial orbit -- RA of Ascending Node undefined'
           node = 0.0d+0
      elseif (vn(2) .gt. 0.0d+0) then
           node = dacos(vn(1)/vnmag)
      else
           node = 2.0d+0 * pi - dacos(vn(1)/vnmag)
      endif
c
c     eccentricity = ecc
c
      rdotv = y(1,i)*y(4,i) + y(2,i)*y(5,i) + y(3,i)*y(6,i)
      f1 = v*v/emu - 1.0d+0/r
      f2 = rdotv/emu
      e(1) = f1*y(1,i) - f2*y(4,i)
      e(2) = f1*y(2,i) - f2*y(5,i)
      e(3) = f1*y(3,i) - f2*y(6,i)
      ecc = dsqrt(e(1)*e(1) + e(2)*e(2) + e(3)*e(3))
c
c     semi-major axis = axis
c
      p = hmag*hmag/emu
      axis = p/(1.0d+0 - ecc*ecc)
c
c     argument of perigee = pgee
c
      edotn = vn(1)*e(1) + vn(2)*e(2) + vn(3)*e(3)
      en = ecc*vnmag
      if (ecc .eq. 0.0d+0) then
           print *,'Circular orbit -- Argument of Perigee undefined'
```

```
            pgee = 0.0d+0
         elseif (vnmag .eq. 0.0d+0) then
            print *,'Equatorial orbit -- Argument of Perigee undefined'
            pgee = 0.0d+0
         elseif (edotn/en .gt. 1.0d+0) then
            pgee = 0.0d+0
         elseif (e(3) .gt. 0.0d+0) then
            pgee = dacos(edotn/en)
         else
            pgee = 2.0d+0 * pi - dacos(edotn/en)
         endif
c
c        true anomaly = true,  mean anomaly = mean
c
         edotr = e(1)*y(1,i) + e(2)*y(2,i) + e(3)*y(3,i)
         er = ecc*r
         if (ecc .eq. 0.0d+0) then
            print *,'Circular orbit -- Mean Anomaly undefined'
            true = 0.0d+0
         elseif (vnmag .eq. 0.0d+0) then
            print *,'Equatorial orbit -- Mean Anomaly undefined'
            true = 0.0d+0
         elseif (edotr/er .gt. 1.0d+0) then
            true = 0.0d+0
         elseif (rdotv .gt. 0.0d+0) then
            true = dacos(edotr/er)
         else
            true = 2.0d+0 * pi - dacos(edotr/er)
         endif
         eanom = 2.0d+0 * datan(dsqrt((1.0d+0 - ecc)/(1.0d+0 + ecc)) *
        #        dtan(true/2.0d+0))
         mean = eanom - ecc*dsin(eanom)
         mean = dmod(mean + 2.0d+0*pi,2.0d+0*pi)
c
c        calculate period = per in seconds
c
         per = (2.0d+0 * pi / dsqrt(emu) * (dsqrt(axis))**3.0d+0)
         permin = per / 60.0d+0
c
         incl = incl * degrad
         node = node * degrad
         pgee = pgee * degrad
         mean = mean * degrad
c
         write (*,900) permin,axis,ecc,incl,node,pgee,mean
c
 900     format(/,2x,'Classical Elements:',//,
        1    2x,'Per    ',f18.13,3x,'minutes',/,
        2   2x,'a     ',f19.13,3x,'km',/,
        3   2x,'e         ',f15.13,/,
        4   2x,'incl    ',f17.13,3x,'degrees',/,
        5   2x,'node    ',f17.13,3x,'degrees',/,
        6   2x,'argp    ',f17.13,3x,'degrees',/,
        7    2x,'manmly ',f18.13,3x,'degrees',/)
c
         return
         end

*******************************************************************************

         subroutine julday(iyyy,mm,id,ih,im,sec, julian)
c
c        year month day hour min sec to modified julian day
c        (minus 2440000.) adapted from Numerical Recipes
c
```

```
      double precision julian
      integer*4 igreg,ijul,ick
c
      igreg = 588829
c
c     get integer part
c
      if(iyyy .lt. 0) iyyy = iyyy + 1
      if( mm .gt. 2) then
          jy = iyyy
          jm = mm+1
      else
          jy = iyyy - 1
          jm = mm + 13
      endif
      ijul = int(365.25*real(jy))+int(30.6001*real(jm))+id+1720995
      ick = id + 31*(mm + 12*iyyy)
      if(ick .ge. igreg) then
           ja = int(0.01*jy)
           ijul = ijul + 2 - ja + int(0.25*real(ja))
      endif
c     correction to modified julian days
      ijul = ijul - 2440000
c
c     fractional part
c
      julian = dble(ijul) -0.5d0 + dble(ih)/24.d0 + dble(im)/1440.d0
     1           + dble(sec)/86400.d0
c
      return
      end

*******************************************************************************

      subroutine caldat( julian, iyyy, mm, id, ih, im, sec )
c
c     convert modified julian day to date and time
c     adapted from numerical recipies
c
      double precision julian, fract
      integer*4 ijul,igreg,ja,jb
c
      igreg = 2299161
c
c     separate integer and fractional part
c
      ijul = idint(julian+0.5) + 2440000
      fract = julian +0.5d0 - dble(ijul - 2440000)
c
c     extract date
c
      if(ijul .ge. igreg) then
          jalpha = int(((ijul-1867216) - 0.25)/36524.25)
          ja = ijul + 1 + jalpha - int(0.25*jalpha)
      else
          ja = ijul
      endif
      jb = ja + 1524
      jc = int(6680.0 + ((jb-2439870)-122.1)/365.25)
      jd = 365*jc + int(0.25*jc)
      je = int((jb - jd)/30.6001)
      id = jb - jd - int(30.6001*je)
      mm = je - 1
      if(mm .gt. 12) mm = mm - 12
      iyyy = jc - 4715
```

```
      if(mm .gt. 2) iyyy = iyyy - 1
      if(iyyy .le. 0) iyyy = iyyy - 1
c
c     extract time
c
      ih = idint( 24.d0*fract )
      fract = fract - dble( ih )/24.d0
      im = idint( 1440.d0*fract )
      fract = fract - dble( im )/1440.d0
      sec = fract*86400.d0
      if(sec .ge. 60.d0) then
           sec = sec - 60.d0
           im = im + 1
      endif
      if(im .ge. 60) then
           im = im - 60
           ih = ih + 1
      endif
      if(ih .ge. 24) then
           ih = ih - 24
           id = id + 1
      endif
c
      return
      end

*******************************************************************************

      subroutine open
c
c     open all files for truth model use
c
      open(3,FILE='input.t')
      open(4,FILE='output.t')
c
      open(16,FILE='iosa.dat')
      open(17,FILE='dgsc.dat')
      open(18,FILE='gtsa.dat')
      open(19,FILE='htsa.dat')
      open(20,FILE='vtsa.dat')
      open(21,FILE='ctsc.dat')
      open(22,FILE='nhsa.dat')
      open(23,FILE='ttsb.dat')
      open(24,FILE='tcsc.dat')
c
      return
      end

*******************************************************************************

      subroutine close
c
c     close all files used in truth model
c
      close(3)
      close(4)
c
      do 100 i = 16,24
           close(i)
 100  continue
c
      return
      end
```

```
*******************************************************************************
          Sensor Locations

Indi  1  -4.671747860   55.477820590   560.500
Reef  2  -7.270030560   72.369998600   -68.375
Guam  3  13.615187820  144.856049380   218.930
Hula  4  21.562265240  201.757894060   429.420
Cook  5  34.822598900  239.498147050   271.530
Pike  6  38.805943055  255.471532222  1899.420
Boss  7  42.947821440  288.373437430   203.370
Pogo  8  76.515364390  291.401141690   147.030
Lion  9  51.117583380  359.093654500   146.590

*******************************************************************************

        Typical Input File

1992 09 10 12 00 00
1992 09 10 13 45 00
15
5097.638 -2716.526 3544.054
5.060657 3.636431 -4.478165
```

Appendix F. Differential Corrector Program Listing

F.1 The Differential Corrector

The differential corrector is a least squares algorithm estimating six orbital elements specified by equinoctial elements. Only two-body orbital motion effects are solved for based on a single track of observation data. The subroutine 'phipphi.for' was written by Dr. Wiesel. The initial reference state is input in rectangular coordinates and converted to equinoctial elements in 'equin.for.' This is after the initial state has been propagated forward using the same integrator subroutines used in the truth model.

```
      program lstsq
c
c     nonlinear least squares algorithm
c
c     least squares common block
c
      common /lst/ x(42)
      double precision x
c
c     hamming common block
c
      common /ham/ t,y(42,4),f(42,4),err(42),dt,mode,n
      double precision t,y,f,err,dt
      integer mode,n,nstep
c
c     constants common block
c
      common /const/ degrad,emu,flat,mu,pi,rearth,tunits,we
      double precision degrad,emu,flat,mu,pi,rearth,tunits,we
c
c     observation storage buffers:
c
      double precision timeob(3000),allobs(3,3000)
      integer iob,nob,idone,iter,irej,ifail,isite(3000)
c
c     internal buffers:
c
c     dtime - (tob-tepoch) in seconds  tepoch - epoch of reference trajectory
c     dx    - correction to the state  tmat   - product of h*phi
c     h     - linearization of data    tob    - time of current observation
c     p     - covariance matrix        ttq1   - T_transpose * Q_inverse
c     phi   - linearization of state   ttq1r  - T_trans*Q_inv*r
c     pinv  - inverse covariance       xref   - reference trajectory
c     q1    - inverse data covariance  z      - actual obs (rho,az,el)
c     r     - residuals                zpred  - predicted observations
c     rms   - root mean square of r's
c
      double precision dt,dtime,dx(6),h(3,6),p(6,6),phi(6,6)
      double precision pinv(6,6),q1(3,3),r(3),reftime,reject,rms(3)
      double precision tepoch,tmat(6,6),tob,ttq1(6,3),ttq1r(6)
      double precision xref(6),z(3),zpred(3)
      integer maxit,iyr,imon,iday,ihr,imin,site
```

```
      real sec
c
c     inverse computational matrices
c
      double precision fac(6,6),b(6),res(6)
      integer ipvt(6)
c
c
c     ****************************************************************
c
c      READ IN INITIAL GUESSES FOR STATE VECTOR & CONTROL PARAMETERS
c
c     ****************************************************************
c
c     constants
c
      degrad = 57.2957795131d+0
      emu = 3.986012d+5
      flat = 1.0d+0/298.257d+0
      pi = 3.141592653589d+0
      rearth = 6378.137d+0
      tunits = 806.8118744d+0
      we = 7.292115856d-5
      mu = emu*(tunits**2/rearth**3)
c
      open(3,FILE='input.d')
      open(4,FILE='output.d')
c
c     read: initial time, reference trajectory guess,
c           max allowed iter, residual reject criteria (# of sigmas)
c
      read (3,*) iyr,imon,iday,ihr,imin,sec
      call julday(iyr,imon,iday,ihr,imin,sec,tepoch)
      read (3,*) (x(ii),ii=1,3)
      read (3,*) (x(ii),ii=4,6)
      read (3,*) maxit
      read (3,*) reject
c
c     ****************************************
c
c       READ IN SITE LOCATION AND OBSERVATIONS
c
c     ****************************************
c
c     convert range to earth radii (er) and degrees to radians
c
      do 100 iob = 1,3000
           read (3,*,END=125) iyr,imon,iday,ihr,imin,sec,
     #                        isite(iob),(allobs(ii,iob),ii=1,3)
           call julday(iyr,imon,iday,ihr,imin,sec,timeob(iob))
           if (iob .eq. 1) reftime = timeob(iob)
           allobs(1,iob) = allobs(1,iob)/rearth
           allobs(2,iob) = allobs(2,iob)/degrad
           allobs(3,iob) = allobs(3,iob)/degrad
 100  continue
c
      write (*,9000)
 9000 format(2x,/,'observation buffer full ... truncated',/)
c
 125  nob = iob - 1
c
c     set the reference trajectory as equinoctial elements at time of 1st ob
c          use hamming to propagate input reference to time of first ob
c
      do 140 i = 1,6
```

```
          y(i,1) = x(i)
 140  continue
c
c     convert t,reftime into seconds for haming routine
c
      t = tepoch * 86400.0d+0
      reftime = reftime * 86400.0d+0
c
c     initialize haming
c          mode = 0 means no phi matrix, EOM only (n = 6 elements)
c          dt = stepsize for hamming in seconds
c
      dt = 15.0d+0
      mode = 0
      n = 6
      nstep = (reftime - t)/dt
c
c     have to worry about possible truncation in calculation of nstep
c
      if (dabs(t + dble(nstep)*dt -reftime) .gt.
     1    dabs(t + dble(nstep)*dt + dt - reftime)) nstep = nstep + 1
c
      nxt = 0
      call haming(nxt)
      if (nxt .eq. 0) then
           print *,'hamming not initializing'
           stop
      endif
c
c     main propagation loop
c
      do 150 i = 1,nstep
           call haming(nxt)
 150  continue
c
c     assign the new reference trajectory back to x
c
      tepoch = t/86400.0d+0
c
      do 160 i = 1,6
           x(i) = y(i,nxt)
 160  continue
c
c     convert reference trajectory to equinoctial elements
c
      call equin
c
      do 175 i = 1,6
           xref(i) = x(i)
 175  continue
c
c     print out input
c
      call caldat(tepoch,iyr,imon,iday,ihr,imin,sec)
      write (4,9010) iyr,imon,iday,ihr,imin,sec,xref,maxit,reject
c
 9010 format(/,25x,'NONLINEAR LEAST SQUARES',//,2x,
     #       'Epoch time:',3x,i4,1x,4(i2,1x),f5.2,//,2x,
     #       'Initial state vector:',f18.9,2(2x,f12.9)
     #                              ,/,23x,f18.9,2(2x,f12.9)//,2x,
     #       'Maximum iterations:',i3,8x,
     #       'Reject if gt ',1p,e9.3,' sigma')
c
c     set last pass flag
c
```

```
      idone = 0
c
c     ************************************************
c
c       BEGIN ITERATION LOOP - NONLINEAR LEAST SQUARES
c
c     ************************************************
c
      do 5000 iter = 1,maxit
c
c          ***************************************
c
c            REINITIALIZE STATE AND PHI MATRIX
c
c          ***************************************
c
c          new reference traj guess is x(1-6)
c          phi ic's x(7-42)
c
           do 200 i = 1,6
                x(i) = xref(i)
 200       continue
           do 205 i = 7,42
                x(i) = 0.0d+0
 205       continue
           do 210 i = 7,42,7
                x(i) = 1.0d+0
 210       continue
c
c          ********************************************************
c
c            INITIALIZE BUFFERS FOR MATRIX PRODUCT ACCUMULATION
c
c          ********************************************************
c
           do 225 i = 1,6
                ttq1r(i) = 0.0d+0
                do 220 j = 1,6
                   pinv(i,j) = 0.0d+0
 220            continue
 225       continue
c
c          reset root mean square value for this iteration
c
           do 250 i = 1,3
                rms(i) = 0.0d+0
 250       continue
c
c          print first or last pass residual headers when necessary
c
           if(iter .eq. 1) write (4,9020)
           if(idone .eq. 1) write (4,9030)
           if((iter .eq. 1) .or. (idone .eq. 1)) write (4,9040)
c
 9020      format(/,2x,'First Pass Residuals: ',/)
 9030      format(/,2x,'Last Pass Residuals:',/)
 9040      format(30x,'Range (er)',4x,
     #            'Azimuth (rad)',3x, 'Elevation (rad)')
c
c
c          *********************************
c
c            OBSERVATION PROCESSING LOOP
c
c          *********************************
```

```
c
      do 1000 iob = 1,nob
c
c           extract this observation
c
            site = isite(iob)
            tob = timeob(iob)
            do 300 i = 1,3
                z(i) = allobs(i,iob)
 300        continue
c
c           ****************************************
c
c             DETERMINE STATE AND PHI AT OB TIME
c
c           ****************************************
c
c           the only time varying element is l = x(1)
c           only phi(1,4) = x(10) element varies
c           phi is stored row by row
c           dtime in tunits
c
            dtime = (tob - tepoch)*(86400.0d+0/tunits)
c
            x(1) = dmod(xref(1) + mu**2*dtime/xref(4)**3,2.0d+0*pi)
            x(10) = -3.0d+0*mu**2*dtime/xref(4)**4
c
c           ****************************************
c
c             OBTAIN MATRICES FOR THIS OBSERVATION
c
c           ****************************************
c
            call obser(tob,site,q1,zpred,h,iter)
c
c           ****************************************
c
c             MATRIX CALCULATIONS - THIS OBSERVATION
c
c           ****************************************
c
c           Form residual vector and test for rejection based on the
c           appropriate sigma value in the q1 matrix and the rejection
c           criteria entered (# of sigmas). Any rejected obs changes the
c           reject flag for the entire set of obs.
c
            irej = 0
            do 310 i = 1,3
                r(i) = z(i) - zpred(i)
                if(dabs(r(i)) .gt. reject/dsqrt(q1(i,i))) irej = 1
 310        continue
c
            if (r(2) .lt. -pi) then
                r(2) = r(2) + 2.0d+0*pi
            elseif (r(2) .gt. pi) then
                r(2) = r(2) - 2.0d+0*pi
            endif
c
c           print residuals for first or last pass only
c
            if (iter .eq. 1 .or. idone .eq. 1) then
                call caldat(tob,iyr,imon,iday,ihr,imin,sec)
                if(irej .eq. 0) then
                    do 350 i = 1,3
                        rms(i) = rms(i) + r(i)**2
```

```
 350                  continue
                      write (4,9100) iyr,imon,iday,
     #                               ihr,imin,sec,(r(ii),ii=1,3)
                 endif
                 if(irej .ne. 0) write (4,9110) iyr,imon,iday,
     #                               ihr,imin,sec,(r(ii),ii=1,3)
               endif
c
 9100          format(2x,i4,1x,4(i2,1x),f5.2,3x,1p
     #               e14.6,2x,e14.6,2x,e14.6)
 9110          format(2x,i4,1x,4(i2,1x),f5.2,3x,1p
     #               e14.6,2x,e14.6,2x,e14.6,3x,' REJECTED')
c
c              skip matrix calculations if this is the last pass
c              (convergence has already been achieved)
c
               if( idone .eq. 1 ) go to 1000
c
c              WAS THIS OBSERVATION REJECTED ?
c
               if(irej .ne. 0) go to 1000
c
c              extract phi matrix in normal form
c
               do 365 i = 1,6
                    do 360 j = 1,6
                         phi(i,j) = x(6*i+j)
 360                 continue
 365           continue
c
c              form matrix product tmat = h * phi
c                                 (3x6) = (3x6)*(6x6)
c
               do 377 i = 1,3
                    do 373 j = 1,6
                         tmat(i,j) = 0.0d+0
                         do 370 k = 1,6
                              tmat(i,j) = tmat(i,j) + h(i,k)*phi(k,j)
 370                     continue
 373                 continue
 377           continue
c
c              form matrix product T_transpose * Q_inverse
c                           (6x3) =  (6x3)         (3x3)
c
               do 387 i = 1,6
                    do 383 j = 1,3
                         ttq1(i,j) = 0.0d+0
                         do 380 k = 1,3
                              ttq1(i,j) = ttq1(i,j) + tmat(k,i)*q1(k,j)
 380                     continue
 383                 continue
 387           continue
c
c              running sum of product (T_trans) (Q_inv) ( T )
c                               (6x6) =       (6x3)      (3x6)
c
               do 397 i = 1,6
                    do 393 j = 1,6
                         do 390 k = 1,3
                              pinv(i,j) = pinv(i,j)+ttq1(i,k)*tmat(k,j)
 390                     continue
 393                 continue
 397           continue
c
```

```
c               running sum of product (T_trans) (Q_inv) ( r )
c                                 (6x1) =       (6x3)      (3x1)
c
                do 405 i = 1,6
                     do 400 j = 1,3
                          ttq1r(i) = ttq1r(i) + ttq1(i,j)*r(j)
 400                 continue
 405            continue
c
c               loop back for more data
c
c          ******************************
c
c            END OF OBS PROCESSING LOOP
c
c          ******************************
c
 1000       continue
c
c          write root mean square values after 1st and last pass
c
           if(iter .eq. 1 .or. idone .eq. 1) then
                do 1010 i = 1,3
                     rms(i) = dsqrt(rms(i)/dble(nob))
 1010           continue
                write (4,9150) (rms(ii),ii=1,3)
 9150           format (/,'Root Mean Square',
     #                1p,11x,e14.6,2x,e14.6,2x,e14.6)
           endif
c
c          have we just finished printing last pass residuals?
c
          if (idone .eq. 1) goto 6000
c
c          ****************************************
c
c            DATA IS PROCESSED ... IMPROVE ESTIMATE
c
c          ****************************************
c
c           write(*,9845) ((pinv(ii,jj),jj=1,6),ii=1,6)
c
c          invert matrix (T_trans) (Q_inv) ( T ) to find covariance P
c
c          IMSL routine taken from Math/Library page 99.
c               two-step routine to compute high accuracy inverse of a real-
c               symmetric matrix by first computing the UDU_trans
c               factorization and then solving the system
c
c          b is the identity matrix - 1 column at a time
c
           do 1050 l = 1,6
                b(l) = 0.0d+0
 1050       continue
c
c          first factor pinv and determine the pivots
c
           call dlftsf(6,pinv,6,fac,6,ipvt)
c
c          next solve the system 1 column at a time and iteratively refine
c
           do 1055 j = 1,6
                b(j) = 1.0d+0
                call dlfisf(6,pinv,6,fac,6,ipvt,b,p(1,j),res)
                b(j) = 0.0d+0
```

```
 1055      continue
c
c          multiply P by (T trans) (Q inv) ( r ) to get correction to state
c                 (6x6)             (6x1)
c
           do 1110 i = 1,6
                dx(i) = 0.0d+0
                do 1100 j = 1,6
                     dx(i) = dx(i) + p(i,j) * ttq1r(j)
 1100           continue
 1110      continue
c
c          *************************
c
c            CHECK FOR CONVERGENCE
c
c          *************************
c
c          if the correction is smaller than 1/100th of the corresponding
c          diagonal element of the covariance matrix, then stop
c
           ifail = 0
           do 1120 i = 1,6
                if(dabs(dx(i)) .gt. 0.01d+0*dsqrt(dabs(p(i,i)))) ifail=1
 1120      continue
c
c          print iteration
c
           write (*,9200) iter,(dx(ii),ii=1,6)
 9200      format(/,2x,'Iteration ',i3,
     #            2x,'state corrections',1p,2(/,e14.6,2(2x,e14.6)))
c
c          add in state corrections to the reference trajectory
c
           do 1130 i = 1,6
                xref(i) = xref(i) + dx(i)
 1130      continue
           xref(1) = dmod(xref(1),2.0d+0*pi)
c
c          print current best guess
c
           write (*,9210) (xref(ii),ii=1,6)
 9210      format(/,2x,
     #        'Current reference trajectory state vector at epoch:',
     #        2(/,f14.9,2(2x,f14.9)))
c
c          have we just converged?
c
           if(ifail .eq. 0 ) idone = 1
           if(idone .eq. 1 ) write (*,9220)
 9220      format(//,2x,'CONVERGENCE ACHIEVED.',/,2x,
     #            'In nominia Gaussiam trajectorum referentia ',
     #            'declarium est estimatia.',/)
c
c          failure, yet again ...
c
c          *************************
c
c            END OF ITERATION LOOP
c
c          *************************
c
 5000 continue
c
c          failure processing ... max iterations exceeded
```

```
c
      write (*,9300)
 9300 format(2x,/,'Maximum iteration limit exceeded.')
c
      go to 8000
c
c     ************************************************************
c
c       !!!!!!!!!!!!!!!!!!!!! SUCCESS !!!!!!!!!!!!!!!!!!!!!!
c
c     ************************************************************
c
c     print covariance matrix
c
 6000 write (4,9310) ((p(ii,jj),jj=1,6),ii=1,6)
 9310 format(/,2x,'Covariance Matrix at epoch is:',/,1p,
     #       6(1x,6(e12.5,1x),/) )
c
c     print state at time of last observation
c
      call caldat(tob,iyyy,imon,iday,ihr,imin,sec)
c
      write (4,9320) iyyy,imon,iday,ihr,imin,sec,(x(ii),ii=1,6)
 9320 format(1x,'state at t = ',i4,1x,4(i2,1x),f5.2,
     #       2(/,f13.9,2(2x,f12.9)))
c
c     calc covariance at last observation time
c
c     extract phi
c
      do 6020 i=1,6
           do 6010 j = 1,6
                phi(i,j) = x(6*i+j)
 6010      continue
 6020 continue
c
c     move covariance (new covariance now stored in pinv)
c
      call phpph(p,phi,pinv)
c
      write (4,9340) ((pinv(ii,jj),jj=1,6),ii=1,6)
 9340 format(/,2x,'Covariance at time of last observation:',/,1p,
     #    6(1x,6(e12.5,1x),/) )
c
 8000 end
c
c
      include 'haming.for'
      include 'equin.for'
      include 'obser.for'
      include 'phipphi.for'
      include 'julian.for'

*******************************************************************************

      subroutine equin
c
c     calculate equinoctial elements (earth radii, radians, time units)
c     from r and v (km and km/sec)
c          this is the same routine as elemnt from truthmdl
c          plus going from classical to equinoctial
c
c     least squares common block
c
      common /lst/ x(42)
```

```
      double precision x
c
c     general double precision statements
c
      double precision axis,ecc,incl,node,pgee,true,eanom,mean
      double precision r,v,h(3),hmag,vn(3),vnmag
      double precision rdotv,f1,f2,e(3),p
      double precision edotn,en,edotr,er
c
c     constants
c
      common /const/ degrad,emu,flat,mu,pi,rearth,tunits,we
      double precision degrad,emu,flat,mu,pi,rearth,tunits,we
c
      r = dsqrt(x(1)*x(1) + x(2)*x(2) + x(3)*x(3))
      v = dsqrt(x(4)*x(4) + x(5)*x(5) + x(6)*x(6))
c
c     angular momentum vector, h = r x v, and magnitude, hmag
c
      h(1) = x(2)*x(6) - x(3)*x(5)
      h(2) = x(3)*x(4) - x(1)*x(6)
      h(3) = x(1)*x(5) - x(2)*x(4)
      hmag = dsqrt(h(1)*h(1) + h(2)*h(2) + h(3)*h(3))
c
c     nodal vector, vn = k x h, and magnitude, vnmag
c
      vn(1) = -h(2)
      vn(2) = h(1)
      vn(3) = 0.0d+0
      vnmag = dsqrt(vn(1)*vn(1) + vn(2)*vn(2) + vn(3)*vn(3))
c
c     inclination = incl
c
      incl = dacos(h(3)/hmag)
c
c     right ascension of ascending node = node
c
      if (vnmag .eq. 0.0d+0) then
           print *,'Equatorial orbit -- RA of Ascending Node undefined'
           node = 0.0d+0
      elseif (vn(2) .gt. 0.0d+0) then
           node = dacos(vn(1)/vnmag)
      else
           node = 2.0d+0 * pi - dacos(vn(1)/vnmag)
      endif
c
c     eccentricity = ecc
c
      rdotv = x(1)*x(4) + x(2)*x(5) + x(3)*x(6)
      f1 = v*v/emu - 1.0d+0/r
      f2 = rdotv/emu
      e(1) = f1*x(1) - f2*x(4)
      e(2) = f1*x(2) - f2*x(5)
      e(3) = f1*x(3) - f2*x(6)
      ecc = dsqrt(e(1)*e(1) + e(2)*e(2) + e(3)*e(3))
c
c     semi-major axis = axis (km)
c
      p = hmag*hmag/emu
      axis = p/(1.0d+0 - ecc*ecc)
c
c     argument of perigee = pgee
c
      edotn = vn(1)*e(1) + vn(2)*e(2) + vn(3)*e(3)
      en = ecc*vnmag
```

```
      if (ecc .eq. 0.0d+0) then
           print *,'Circular orbit -- Argument of Perigee undefined'
           pgee = 0.0d+0
      elseif (vnmag .eq. 0.0d+0) then
           print *,'Equatorial orbit -- Argument of Perigee undefined'
           pgee = 0.0d+0
      elseif (edotn/en .gt. 1.0d+0) then
           pgee = 0.0d+0
      elseif (e(3) .gt. 0.0d+0) then
           pgee = dacos(edotn/en)
      else
           pgee = 2.0d+0 * pi - dacos(edotn/en)
      endif
c
c     true anomaly = true,  mean anomaly = mean
c
      edotr = e(1)*x(1) + e(2)*x(2) + e(3)*x(3)
      er = ecc*r
      if (ecc .eq. 0.0d+0) then
           print *,'Circular orbit -- Mean Anomaly undefined'
           true = 0.0d+0
      elseif (vnmag .eq. 0.0d+0) then
           print *,'Equatorial orbit -- Mean Anomaly undefined'
           true = 0.0d+0
      elseif (edotr/er .gt. 1.0d+0) then
           true = 0.0d+0
      elseif (rdotv .gt. 0.0d+0) then
           true = dacos(edotr/er)
      else
           true = 2.0d+0 * pi - dacos(edotr/er)
      endif
      eanom = 2.0d+0 * datan(dsqrt((1.0d+0 - ecc)/(1.0d+0 + ecc)) *
     $        dtan(true/2.0d+0))
      mean = eanom - ecc*dsin(eanom)
c
c     calculate the equinoctial elements
c         x(4) converted from km**2/sec to er**2/sec
c
      x(1) = mean
      x(2) = ecc * dcos(pgee)
      x(3) = dtan(incl/2.0d+0) * dcos(node)
      x(4) = dsqrt(emu*axis)*(tunits/rearth**2)
      x(5) = ecc * dsin(pgee)
      x(6) = dtan(incl/2.0d+0) * dsin(node)
c
      write (*,900) (x(ii),ii=1,6)
c
 900  format(/,2x,'Equinoctial Elements:',//,
     1   2x,'l    ',f17.13,2x,'radians',/,
     2   2x,'k    ',f17.13,/,
     3   2x,'psi  ',f17.13,3x,/,
     4   2x,'capl ',f17.13,2x,'DU**2/TU',/,
     5   2x,'h    ',f17.13,3x,/,
     6    2x,'chi  ',f17.13,3x,/)
c
      return
      end

*******************************************************************************

      subroutine obser (tob,isite,q1,zpred,h,iter)
c
c     This routine establishes the data covariance, computes the
c     predicted range, azimuth, elevation for the current equinoctial
c     state vector (in call to antnna), and computes the data linearization
```

```
c     matrix h thru extensive use of partial derivatives and the chain rule.
c
      common /lst/ x(42)
      double precision x
c
      double precision tob,q1(3,3),zpred(3),h(3,6)
      integer isite
c
c     declarations for predicted data vector
c
      double precision axis,ecc,incl,node,pgee,mean
      double precision error,anom0,anom1,eanom,cnu,snu,rmag,p
      double precision pqw(6),rho(6),rot(3,3),eci(6)
c
c     declarations for h partial derivatives
c
      double precision phi,theta,danom(6),dnu(6),drmag(6)
      double precision drot11(6),drot12(6),drot21(6),drot22(6)
      double precision drot31(6),drot32(6),dr(6,6),root,droot(6)
      double precision equin(6,6),ecieqn(6,6),drho(3,3),geci(3,6)
c
c     constants
c
      common /const/ degrad,emu,flat,mu,pi,rearth,tunits,we
      double precision degrad,emu,flat,mu,pi,rearth,tunits,we
c
c     *********************
c
c       Q INVERSE MATRIX
c
c     *********************
c
c     sigma = 100 m in range = 1.567856e-5 earth radii,
c           .025 deg = .000436 radians in az and el
c     diagonal elements are 1/sigma**2
c
      q1(1,1) = 4.0681d+9
      q1(2,2) = 5.2525d+6
      q1(3,3) = 5.2525d+6
      q1(1,2) = 0.0d+0
      q1(1,3) = 0.0d+0
      q1(2,1) = 0.0d+0
      q1(2,3) = 0.0d+0
      q1(3,1) = 0.0d+0
      q1(3,2) = 0.0d+0
c
c     *********************************
c
c       PREDICTED DATA VECTOR - ZPRED
c
c     *********************************
c
c     predicted data comes from the current equinoctial element set
c     converted to range, azimuth, elevation
c
c     first determine the classical elements:
c          axis(in DU),ecc,incl,node,pgee,mean
c
      axis = x(4)**2/mu
      ecc = dsqrt(x(2)*x(2) + x(5)*x(5))
      incl = 2.0d+0 * datan(dsqrt(x(3)*x(3)+x(6)*x(6)))
      if (x(3) .lt. 0.0d+0) then
           node = pi - dasin(x(6)/dsqrt(x(3)*x(3)+x(6)*x(6)))
      else
           if (x(6) .gt. 0.0d+0) then
```

```
                node = dasin(x(6)/dsqrt(x(3)*x(3)+x(6)*x(6)))
            else
                node = 2.0d+0*pi +
     #                 dasin(x(6)/dsqrt(x(3)*x(3)+x(6)*x(6)))
            endif
        endif
        if (x(2) .lt. 0.0d+0) then
      pgee = pi - dasin(x(5)/dsqrt(x(2)*x(2)+x(5)*x(5)))
        else
            if (x(5) .gt. 0.0d+0) then
                pgee = dasin(x(5)/dsqrt(x(2)*x(2)+x(5)*x(5)))
            else
                pgee = 2.0d+0*pi +
     #                 dasin(x(5)/dsqrt(x(2)*x(2)+x(5)*x(5)))
            endif
        endif
        mean = x(1)
c
c       error check
c
        if (ecc .ge. 1.0d+0) then
            print *
            print *,'Diverged'
            print *
            stop
        endif
c
c       iterate to find eccentric anomaly (BM&W page 222)
c
        error = 1.0d+0
        anom1 = mean
 100    anom0 = anom1
        anom1 = anom0 + (mean-(anom0-ecc*dsin(anom0)))/
     #                  (1.0d+0-ecc*dcos(anom0))
        if (anom1 .eq. 0.0d+0) then
            error = 0.0d+0
        else
            error = dabs((anom1-anom0)/anom1)
        endif
        if (error .gt. 1.0d-15) go to 100
        eanom = anom1
c
c       now need cos(nu) and sin(nu)  (BM&W page 187)
c
        cnu = (ecc-dcos(eanom))/(ecc*dcos(eanom)-1.0d+0)
        rmag = axis*(1.0d+0 - ecc * dcos(eanom))
        snu = (axis*dsqrt(1.0d+0-ecc*ecc)*dsin(eanom))/rmag
c
c       next find r and v in the PQW frame  (BM&W page 72-3)
c
        p = axis*(1.0d+0 - ecc*ecc)
        pqw(1) = rmag*cnu
        pqw(2) = rmag*snu
        pqw(3) = 0.0d+0
        pqw(4) = -dsqrt(mu/p)*snu
        pqw(5) = dsqrt(mu/p)*(ecc+cnu)
        pqw(6) = 0.0d+0
c
c       rotate to ECI  (rotation matrix from BM&W page 82-3)
c
        rot(1,1) = dcos(node)*dcos(pgee)-dsin(node)*dsin(pgee)*dcos(incl)
        rot(1,2) = -dcos(node)*dsin(pgee)-dsin(node)*dcos(pgee)*dcos(incl)
        rot(2,1) = dsin(node)*dcos(pgee)+dcos(node)*dsin(pgee)*dcos(incl)
        rot(2,2) = -dsin(node)*dsin(pgee)+dcos(node)*dcos(pgee)*dcos(incl)
        rot(3,1) = dsin(pgee)*dsin(incl)
```

```
      rot(3,2) = dcos(pgee)*dsin(incl)
c
      eci(1) = rot(1,1)*pqw(1) + rot(1,2)*pqw(2)
      eci(2) = rot(2,1)*pqw(1) + rot(2,2)*pqw(2)
      eci(3) = rot(3,1)*pqw(1) + rot(3,2)*pqw(2)
      eci(4) = rot(1,1)*pqw(4) + rot(1,2)*pqw(5)
      eci(5) = rot(2,1)*pqw(4) + rot(2,2)*pqw(5)
      eci(6) = rot(3,1)*pqw(4) + rot(3,2)*pqw(5)
c
      call antnna(isite,tob,eci,zpred,rho,phi,theta)
c
c     **********************
c
c       CALCULATE H MATRIX
c
c     **********************
c
c     Step 1 - Define partials of Eccentric Anomaly from M = E - e*sin(E)
c              with respect to the classical elements (a,e,i,node,pgee,mean).
c
      danom(1) = 0.0d+0
      danom(2) = dsin(eanom)/(1.0d+0 - ecc*dcos(eanom))
      danom(3) = 0.0d+0
      danom(4) = 0.0d+0
      danom(5) = 0.0d+0
      danom(6) = 1.0d+0/(1.0d+0 - ecc*dcos(eanom))
c
c     Step 2 - Define partials of True Anomaly from
c              nu = 2*atan(sqrt((1+e)/(1-e))*tan(eanom/2))
c              with respect to the classical elements.
c
      dnu(1) = 0.0d+0
      dnu(2) = 2.0d+0*(danom(2)*dsqrt((1.0d+0 + ecc)/
     #         (1.0d+0 - ecc))/(2.0d+0*dcos(eanom/2.0d+0)**2) +
     #         dtan(eanom/2.0d+0)/((1.0d+0 - ecc)**2
     #         *dsqrt((1.0d+0 + ecc)/(1.0d+0 - ecc))))/
     #         (1.0d+0 + (1.0d+0 + ecc)*dtan(eanom/2.0d+0)**2/
     #         (1.0d+0 - ecc))
      dnu(3) = 0.0d+0
      dnu(4) = 0.0d+0
      dnu(5) = 0.0d+0
      dnu(6) = danom(6)**2 * (1.0d+0 - ecc) *
     #         dsqrt((1.0d+0 +ecc)/(1.0d+0 - ecc))
c
c     Step 3 - Define partials of rmag from rmag = a(1-e*cos(E))
c              with respect to the classical elements.
c
      drmag(1) = (1.0d+0 - ecc*dcos(eanom))
      drmag(2) = axis*(ecc*dsin(eanom)*danom(2)-dcos(eanom))
      drmag(3) = 0.0d+0
      drmag(4) = 0.0d+0
      drmag(5) = 0.0d+0
      drmag(6) = axis*ecc*dsin(eanom)*danom(6)
c
c     Step 4 - Define partials of the rotation matrix - rot(3,3)
c              with respect to the classical elements.
c
      drot11(1) = 0.0d+0
      drot11(2) = 0.0d+0
      drot11(3) = dsin(node)*dsin(pgee)*dsin(incl)
      drot11(4) = -rot(2,1)
      drot11(5) = rot(1,2)
      drot11(6) = 0.0d+0
c
      drot12(1) = 0.0d+0
```

```
      drot12(2) = 0.0d+0
      drot12(3) = dsin(node)*dcos(pgee)*dsin(incl)
      drot12(4) = -rot(2,2)
      drot12(5) = -rot(1,1)
      drot12(6) = 0.0d+0
c
      drot21(1) = 0.0d+0
      drot21(2) = 0.0d+0
      drot21(3) = -dcos(node)*dsin(pgee)*dsin(incl)
      drot21(4) = rot(1,1)
      drot21(5) = rot(2,2)
      drot21(6) = 0.0d+0
c
      drot22(1) = 0.0d+0
      drot22(2) = 0.0d+0
      drot22(3) = -dcos(node)*dcos(pgee)*dsin(incl)
      drot22(4) = rot(1,2)
      drot22(5) = -rot(2,1)
      drot22(6) = 0.0d+0
c
      drot31(1) = 0.0d+0
      drot31(2) = 0.0d+0
      drot31(3) = dsin(pgee)*dcos(incl)
      drot31(4) = 0.0d+0
      drot31(5) = rot(3,2)
      drot31(6) = 0.0d+0
c
      drot32(1) = 0.0d+0
      drot32(2) = 0.0d+0
      drot32(3) = dcos(pgee)*dcos(incl)
      drot32(4) = 0.0d+0
      drot32(5) = -rot(3,1)
      drot32(6) = 0.0d+0
c
c     Step 5 - Define partials of position and velocity from
c              r_eci = rot * r_pqw = rot * (r*cos(nu) + r*sin(nu))
c              v_eci = rot * (-sin(nu)*sqrt(mu/a(1-e^2)) + (e+cos(nu)) * sqrt)
c              with respect to the classical elements.
c     Completion of this step achieves the partials of the eci position and
c     and velocity vectors with respect to the classical elements.
c
      dr(1,1) = rot(1,1)*cnu*drmag(1) + rot(1,2)*snu*drmag(1)
      dr(1,2) = -rot(1,1)*rmag*snu*dnu(2) + rot(1,1)*cnu*drmag(2) +
     #           rot(1,2)*rmag*cnu*dnu(2) + rot(1,2)*snu*drmag(2)
      dr(1,3) = rmag*cnu*drot11(3) + rmag*snu*drot12(3)
      dr(1,4) = rmag*cnu*drot11(4) + rmag*snu*drot12(4)
      dr(1,5) = rmag*cnu*drot11(5) + rmag*snu*drot12(5)
      dr(1,6) = -rot(1,1)*rmag*snu*dnu(6) + rot(1,1)*cnu*drmag(6) +
     #           rot(1,2)*rmag*cnu*dnu(6) + rot(1,2)*snu*drmag(6)
c
      dr(2,1) = rot(2,1)*cnu*drmag(1) + rot(2,2)*snu*drmag(1)
      dr(2,2) = -rot(2,1)*rmag*snu*dnu(2) + rot(2,1)*cnu*drmag(2) +
     #           rot(2,2)*rmag*cnu*dnu(2) + rot(2,2)*snu*drmag(2)
      dr(2,3) = rmag*cnu*drot21(3) + rmag*snu*drot22(3)
      dr(2,4) = rmag*cnu*drot21(4) + rmag*snu*drot22(4)
      dr(2,5) = rmag*cnu*drot21(5) + rmag*snu*drot22(5)
      dr(2,6) = -rot(2,1)*rmag*snu*dnu(6) + rot(2,1)*cnu*drmag(6) +
     #           rot(2,2)*rmag*cnu*dnu(6) + rot(2,2)*snu*drmag(6)
c
      dr(3,1) = rot(3,1)*cnu*drmag(1) + rot(3,2)*snu*drmag(1)
      dr(3,2) = -rot(3,1)*rmag*snu*dnu(2) + rot(3,1)*cnu*drmag(2) +
     #           rot(3,2)*rmag*cnu*dnu(2) + rot(3,2)*snu*drmag(2)
      dr(3,3) = rmag*cnu*drot31(3) + rmag*snu*drot32(3)
      dr(3,4) = 0.0d+0
      dr(3,5) = rmag*cnu*drot31(5) + rmag*snu*drot32(5)
```

```
      dr(3,6) = -rot(3,1)*rmag*snu*dnu(6) + rot(3,1)*cnu*drmag(6) +
     #          rot(3,2)*rmag*cnu*dnu(6) + rot(3,2)*snu*drmag(6)
c
c     Intermediate step to define partials of the square root term in each
c     velocity component equation.  Root = sqrt(mu/a*(1-e**2)).
c
      root = dsqrt(mu/(axis*(1.0d+0 - ecc*ecc)))
      droot(1) = -root/(2.0d+0 * axis)
      droot(2) = ecc*root/(1.0d+0 - ecc*ecc)
      droot(3) = 0.0d+0
      droot(4) = 0.0d+0
      droot(5) = 0.0d+0
      droot(6) = 0.0d+0
c
      dr(4,1) = -rot(1,1)*snu*droot(1) + rot(1,2)*(ecc+cnu)*droot(1)
      dr(4,2) = -rot(1,1)*root*cnu*dnu(2) - rot(1,1)*snu*droot(2) +
     #          rot(1,2)*root*(1.0d+0 - snu*dnu(2)) +
     #          rot(1,2)*(ecc+cnu)*droot(2)
      dr(4,3) = -root*snu*drot11(3) + root*(ecc+cnu)*drot12(3)
      dr(4,4) = -root*snu*drot11(4) + root*(ecc+cnu)*drot12(4)
      dr(4,5) = -root*snu*drot11(5) + root*(ecc+cnu)*drot12(5)
      dr(4,6) = -rot(1,1)*root*cnu*dnu(6) - rot(1,2)*root*snu*dnu(6)
c
      dr(5,1) = -rot(2,1)*snu*droot(1) + rot(2,2)*(ecc+cnu)*droot(1)
      dr(5,2) = -rot(2,1)*root*cnu*dnu(2) - rot(2,1)*snu*droot(2) +
     #          rot(2,2)*root*(1.0d+0 - snu*dnu(2)) +
     #          rot(2,2)*(ecc+cnu)*droot(2)
      dr(5,3) = -root*snu*drot21(3) + root*(ecc+cnu)*drot22(3)
      dr(5,4) = -root*snu*drot21(4) + root*(ecc+cnu)*drot22(4)
      dr(5,5) = -root*snu*drot21(5) + root*(ecc+cnu)*drot22(5)
      dr(5,6) = -rot(2,1)*root*cnu*dnu(6) - rot(2,2)*root*snu*dnu(6)
c
      dr(6,1) = -rot(3,1)*snu*droot(1) + rot(3,2)*(ecc+cnu)*droot(1)
      dr(6,2) = -rot(3,1)*root*cnu*dnu(2) - rot(3,1)*snu*droot(2) +
     #          rot(3,2)*root*(1.0d+0 - snu*dnu(2)) +
     #          rot(3,2)*(ecc+cnu)*droot(2)
      dr(6,3) = -root*snu*drot31(3) + root*(ecc+cnu)*drot32(3)
      dr(6,4) = 0.0d+0
      dr(6,5) = -root*snu*drot31(5) + root*(ecc+cnu)*drot32(5)
      dr(6,6) = -rot(3,1)*root*cnu*dnu(6) - rot(3,2)*root*snu*dnu(6)
c
c     Step 6 - Define partials of classical elements with respect to
c              the equinoctial elements using the relationships between
c              the elements as defined in the predicted data vector section
c              above.
c
      equin(1,1) = 0.0d+0
      equin(1,2) = 0.0d+0
      equin(1,3) = 0.0d+0
      equin(1,4) = 2.0d+0 * x(4)/mu
      equin(1,5) = 0.0d+0
      equin(1,6) = 0.0d+0
c
      equin(2,1) = 0.0d+0
      equin(2,2) = x(2)/ecc
      equin(2,3) = 0.0d+0
      equin(2,4) = 0.0d+0
      equin(2,5) = x(5)/ecc
      equin(2,6) = 0.0d+0
c
      equin(3,1) = 0.0d+0
      equin(3,2) = 0.0d+0
      equin(3,3) = (2.0d+0*x(3))/(dsqrt(x(3)*x(3)+x(6)*x(6))*
     #             (1.0d+0 + x(3)*x(3) + x(6)*x(6)))
      equin(3,4) = 0.0d+0
```

```
      equin(3,5) = 0.0d+0
      equin(3,6) = (2.0d+0*x(6))/(dsqrt(x(3)*x(3)+x(6)*x(6))*
     #             (1.0d+0 + x(3)*x(3) + x(6)*x(6)))
c
      equin(4,1) = 0.0d+0
      equin(4,2) = 0.0d+0
      equin(4,3) = -x(6)/(x(3)*x(3) + x(6)*x(6))
      equin(4,4) = 0.0d+0
      equin(4,5) = 0.0d+0
      equin(4,6) = x(3)/(x(3)*x(3) + x(6)*x(6))
c
      equin(5,1) = 0.0d+0
      equin(5,2) = -x(5)/(ecc*ecc)
      equin(5,3) = 0.0d+0
      equin(5,4) = 0.0d+0
      equin(5,5) = x(2)/(ecc*ecc)
      equin(5,6) = 0.0d+0
c
      equin(6,1) = 1.0d+0
      equin(6,2) = 0.0d+0
      equin(6,3) = 0.0d+0
      equin(6,4) = 0.0d+0
      equin(6,5) = 0.0d+0
      equin(6,6) = 0.0d+0
c
c     Step 7 - Multiply two matrices together.  The matrices are
c              partial x_eci/partial x_class  X  partial x_class/partial x_equin
c              giving partial x_eci/partial x_equinoctial as a 6 X 6 matrix.
c     Completion of this step gives 2/3 of the h matrix.
c
      do 530 i = 1,6
           do 520 j = 1,6
                ecieqn(i,j) = 0.0d+0
                do 510 k = 1,6
                     ecieqn(i,j) = ecieqn(i,j) + dr(i,k)*equin(k,j)
 510            continue
 520       continue
 530  continue
c
c
c     Step 8 - Compute partials of the data vector (range, azimuth, elevation)
c              with respect to the eci position vector.  Multiplying this 3 X 6
c              matrix with the 6 X 6 ecieqn matrix gives the h matrix.
c
c     First need the partials of the topocentric range with respect to the
c     eci position.
c
      drhor(1,1) = dsin(phi) * dcos(theta)
      drhor(1,2) = dsin(phi) * dsin(theta)
      drhor(1,3) = -dcos(phi)
      drhor(2,1) = -dsin(theta)
      drhor(2,2) = dcos(theta)
      drhor(2,3) = 0.0d+0
      drhor(3,1) = cos(phi) * cos(theta)
      drhor(3,2) = cos(phi) * sin(theta)
      drhor(3,3) = sin(phi)
c
      geci(1,1) = (rho(4)*drhor(1,1) + rho(5)*drhor(2,1) +
     #             rho(6)*drhor(3,1))/zpred(1)
      geci(1,2) = (rho(4)*drhor(1,2) + rho(5)*drhor(2,2) +
     #             rho(6)*drhor(3,2))/zpred(1)
      geci(1,3) = (rho(4)*drhor(1,3) + rho(5)*drhor(2,3) +
     #             rho(6)*drhor(3,3))/zpred(1)
      geci(1,4) = 0.0d+0
      geci(1,5) = 0.0d+0
```

```
      geci(1,6) = 0.0d+0
c
      geci(2,1) = (rho(5)*drhor(1,1) - rho(4)*drhor(2,1))/
     #             (rho(4)*rho(4) + rho(5)*rho(5))
      geci(2,2) = (rho(5)*drhor(1,2) - rho(4)*drhor(2,2))/
     #             (rho(4)*rho(4) + rho(5)*rho(5))
      geci(2,3) = (rho(5)*drhor(1,3) - rho(4)*drhor(2,3))/
     #             (rho(4)*rho(4) + rho(5)*rho(5))
      geci(2,4) = 0.0d+0
      geci(2,5) = 0.0d+0
      geci(2,6) = 0.0d+0
c
      geci(3,1) = (zpred(1)*drhor(3,1) - rho(6)*geci(1,1))/
     #            (zpred(1)*dsqrt(zpred(1)**2 - rho(6)**2))
      geci(3,2) = (zpred(1)*drhor(3,2) - rho(6)*geci(1,2))/
     #            (zpred(1)*dsqrt(zpred(1)**2 - rho(6)**2))
      geci(3,3) = (zpred(1)*drhor(3,3) - rho(6)*geci(1,3))/
     #            (zpred(1)*dsqrt(zpred(1)**2 - rho(6)**2))
      geci(3,4) = 0.0d+0
      geci(3,5) = 0.0d+0
      geci(3,6) = 0.0d+0
c
c     And finally, h ...
c
      do 630 i = 1,3
           do 620 j = 1,6
                h(i,j) = 0.0d+0
                do 610 k = 1,6
                h(i,j) = h(i,j) + geci(i,k)*ecieqn(k,j)
 610            continue
 620       continue
 630  continue
c
      return
      end
c
      include 'antnna.for'

*******************************************************************************

      subroutine antnna(isite,tob,eci,zpred,rho,phi,theta)
c
c     This routine calculates the predicted range, azimuth, elevation
c     in topocentric coordinate frame from a given eci position
c     vector and stores in zpred.  This is for a specific station location
c     rsite, phi, and theta.
c
      double precision tob,eci(3),zpred(3),rho(6),rsite(3)
      double precision ut,jd,tu,gmst,thetag
      double precision phi,lon,h,theta,c,s
      integer isite,iloc
      character*4,site
c
c     constants
c
      common /const/ degrad,emu,flat,mu,pi,rearth,tunits,we
      double precision degrad,emu,flat,mu,pi,rearth,tunits,we
c
c     read in site identifier, phi and lon in degrees (immediately
c     converted to radians), and altitude in meters (converted to a
c     fraction of rearth - DU)
c
      open(15,FILE='sensor.loc',STATUS='OLD')
c
c     Reference: 1992 Astronomical Almanac, page B6
```

```
c
      jd = tob + 2440000.0d+0
      ut = dmod(jd + 0.5d+0,1)
      jd = jd - ut
      tu = (jd-2451545.0d+0)/36525.0d+0
      gmst = 24110.54841d+0 + tu*(8640184.812866d+0 + tu*(0.093104d+0 -
     #       tu*6.2d-6))
      gmst = dmod(gmst + 86400.0d+0*(86400.0d+0*we/(2.0d+0*pi))*ut,
     #            86400.0d+0)
      thetag = 2.0d+0*pi*gmst/86400.0d+0
c
c     control loop for each of nine stations
c          1 = Indi  4 = Hula  7 = Boss
c          2 = Reef  5 = Cook  8 = Pogo
c          3 = Guam  6 = Pike  9 = Lion
c
c     input file contains phi and lon in degrees and h in meters
c     degrees converted to radians and meters to earth radii
c
      do 100 i = 1,9
           read (15,*) site,iloc,phi,lon,h
           if (iloc .eq. isite) then
                phi = phi/degrad
                lon = lon/degrad
                h = h/(1000.0d+0*rearth)
                goto 110
           endif
 100  continue
c
c     Convert longitude and time to Local Sidereal Time
c     Reference:  BM & W page 100
c
 110  theta = dmod(lon + thetag,2.0d+0*pi)
c
c     Convert Geodetic Latitude to Geocentric Latitude
c     Reference:  1992 Astronautical Almanac  page K11
c
      c = 1.0d+0/dsqrt(1.0d+0 + flat*(flat - 2.0d+0)*dsin(phi)**2)
      s = (1.0d+0 - flat)**2 * c
c
c     compute site position
c
      rsite(1) = (c + h)*dcos(phi)*dcos(theta)
      rsite(2) = (c + h)*dcos(phi)*dsin(theta)
      rsite(3) = (s + h)*dsin(phi)
c
c     rho(1-3) are the components of range - inertial
c     rho(4-6) are topocentric
c     Reference:  BM & W page 79
c
      do 10 l = 1,3
           rho(l) = eci(l) - rsite(l)
 10   continue
c
      rho(4) = dsin(phi)*dcos(theta)*rho(1) +
     #         dsin(phi)*dsin(theta)*rho(2) -
     #         dcos(phi)*rho(3)
      rho(5) = -dsin(theta)*rho(1) + dcos(theta)*rho(2)
      rho(6) = dcos(phi)*dcos(theta)*rho(1) +
     #         dcos(phi)*dsin(theta)*rho(2) +
     #         dsin(phi)*rho(3)
c
c     calculate the range, azimuth, and elevation
c     Reference:  BM & W page 85
c
```

```
      zpred(1) = dsqrt(rho(4)**2 + rho(5)**2 + rho(6)**2)
c
      if (rho(4) .gt. 0.0d+0) then
           zpred(2) = pi + datan(-rho(5)/rho(4))
      elseif (rho(5) .lt. 0.0d+0) then
           zpred(2) = 2.0d+0*pi + datan(-rho(5)/rho(4))
      else
           zpred(2) = datan(-rho(5)/rho(4))
      endif
c
      zpred(3) = dasin(rho(6)/zpred(1))
c
      close(15)
c
      return
      end

*******************************************************************************

      subroutine phpph(pin, phi, pout)
c
c     performs covariance propagation phi * p * phi transpose
c
c     dimensions
c
      double precision pin(6,6),phi(6,6),pout(6,6),phip(6,6)
c
c     matrix product phi * p
c
      do 420 i = 1,6
           do 410 j = 1,6
                phip(i,j) = 0.0d+0
                do 400 k = 1,6
                     phip(i,j) = phip(i,j) + phi(i,k)*pin(k,j)
 400            continue
 410       continue
 420  continue
c
c     matrix product p(tf) = phi p phi transpose
c
      do 520 i = 1,6
           do 510 j = 1,6
                pout(i,j) = 0.0d+0
                do 500 k = 1,6
                     pout(i,j) = pout(i,j) + phip(i,k)*phi(j,k)
 500            continue
 510       continue
 520  continue
c
      return
      end

*******************************************************************************
         Typical Input File

1992 09 10 12 00 00
5097.638 -2716.526 3544.054
5.060657 3.636431 -4.478165
15
3.0d+5

 1992  9 10 13 32 30.00   3   2236.466802   342.941839     0.578254
 1992  9 10 13 32 45.00   3   2142.764210   344.154938     1.483839
 1992  9 10 13 33  0.00   3   2049.936953   345.521691     2.424605
 1992  9 10 13 33 15.00   3   1958.161107   346.984290     3.354736
```

Bibliography

1. Bate, Roger R. et al. *Fundamentals of Astrodynamics.* New York: Dover Publications, Inc., 1971.
2. Battin, Richard H. *An Introduction to the Mathematics and Methods of Astrodynamics.* New York: American Institute of Aeronautics and Astronautics, Inc., 1987.
3. Kelso, Thomas S. Pascal program function 'Density.' School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 6 February 1987.
4. Kelso, Thomas S. Pascal program function 'ThetaG_JD.' School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 3 September 1992.
5. Kelso, Thomas S. Pascal program procedure 'Calculate_User_PosVel.' School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 6 September 1992.
6. Linnik, Yu. V. *Method of Least Squares and Principles of the Theory of Observations.* New York: Pergamon Press, 1961.
7. Minka, Karlis. *Orbit Determination and Analysis by the Minimum Variance Method.* Martin Company, AFCRL-65-579. Contract AF19(628)-4167. Bedford MA: Air Force Cambridge Research Laboratories, August 1965.
8. Nautical Almanac Office, U. S. Naval Observatory. *The Astronomical Almanac for the Year 1992.* Washington: U. S. Government Printing Office, 1991.
9. North American Aerospace Defense Command. *Mathematical Foundation for Space Computational Center Astrodynamic Theory.* Technical Publication SCC 008. Colorado Springs, Colorado: HQ AFSPACECOM, 6 April 1982.
10. Raol, J. R. and N. K. Sinha. "On the Orbit Determination Problem," *IEEE Transactions on Aerospace and Electronic Systems, AES:* 274-290 (May 1985).
11. Schaaf, William L. *Carl Friedrich Gauss: Prince of Mathematicians.* New York: Franklin Watts, Inc., 1964.
12. USSPACECOM Space Surveillance Center. *Space Surveillance Center Orbital Analysis Officer Positional Handbook* (Version 1.5C). Colorado Springs, Colorado: HQ AFSPACECOM, March 1989.
13. National Aeronautics and Space Administration. *The 1976 Standard Atmosphere Above 86-km Altitude.* NASA SP-398. Washington: 1976.
14. Wiesel, William E., Jr. Class handout distributed in MECH 636, Advanced Astrodynamics. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, January 1992.
15. Wiesel, William E., Jr. Class handout distributed in MECH 731, Modern Methods of Orbit Determination. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1992.

Vita

Captain Michael S. Wasson was born on 30 October 1965 in Marquette, Michigan. He graduated from Negaunee High School in Negaunee, Michigan in 1983 and attended the University of Tampa, Florida, graduating with a Bachelor of Science in Mathematics and Computer Science in May, 1987. He graduated Magna Cum Laude and ROTC Distinguished Graduate. On 30 April 1987, he received a regular commission in the USAF and on 23 May 1987 he married the former Amy Lynne Phelps. He began his first tour in the Air Force at Undergraduate Space Training (UST) at Lowry AFB, Colorado in November 1987. After graduating from UST in March 1988, he began training as an orbit analyst for the 1st Satellite Control Squadron at Falcon AFB, Colorado and served in that position from June 1988 to May 1991 predicting orbits and planning maneuvers for GPS, DSP, and DMSP. While an orbit analyst, he was responsible for documentation, scheduling, and for 1-1/2 years served as chief of training for orbit analysts. In May, 1991 he entered the Space Operations program in the School of Engineering, Air Force Institute of Technology, specializing in Advanced Astrodynamics.

Permanent address: 596 M-35
Negaunee, Mi 49866

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
DATA REDUCTION WITH LEAST SQUARES DIFFERENTIAL CORRECTION USING EQUINOCTIAL ELEMENTS

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael S. Wasson, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GA/ENY/92D-15

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This study investigates earth satellite orbit estimation on a track of range, azimuth, and elevation data from a single tracking station. The estimation routine is a least squares batch filter based solely on two-body orbital motion. Using equinoctial elements for the reference orbit avoids the numerical difficulties of the classical elements at eccentricities near zero and inclinations near zero or 90 degrees. Orbits for Mir, DMSP, Explorer, Cosmos, and GPS are investigated. The goal of this study is to reduce orbit information from observations (range, azimuth, and elevation) to an element set and a covariance matrix without considering perturbation effects. The results indicate that the lower orbiting earth satellites had large J_2 perturbations on the equinoctial elements causing the differential corrector to diverge. Higher orbiting satellites had minimal J_2 effects and the correction process sufficiently extracted all information from the data and successfully reduced the observations to an element set and a covariance matrix.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Equinoctial Elements; Differential Correction; Least Squares Method; Data Reduction; Batch Processing; Earth Orbits; Two-Body Motion	146
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL